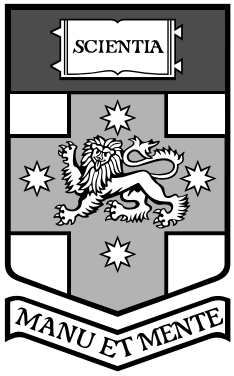
THE UNIVERSITY OF NEW SOUTH WALES,
SCHOOL OF ELECTRICAL ENGINEERING & TELECOMMUNICATIONS
SCHOOL OF COMPUTER SCIENCE & ENGINEERING

Hardware and Software Infrastructure for the Optimisation of Sunswift II

A THESIS SUBMITTED FOR THE DEGREE OF
BACHELOR OF ENGINEERING



David Snowdon, 2234390

November, 2002

Supervisors: Gernot Heiser, Jeff Cotter

To Jimmy Smart and James Ascott

Abstract

Sunswift II is the UNSW entrant in the World Solar Challenge, a solar car race that runs from Darwin to Adelaide. It is a complex amalgamation of mechanical and electrical systems where reliability and efficiency are of the utmost importance.

This thesis outlines the design and development of an integrated telemetry and control system for Sunswift II which provides significantly improved reliability, expandability and flexibility compared with the ad-hoc, limited system previously implemented. As part of this development, the car's maximum power point trackers are examined, and the software redesigned in order to solve many issues that were evident during a previous event — a significant reliability improvement is observed.

Acknowledgements

A number of people, businesses, and organisations have contributed to this work.

Firstly, thanks to my supervisors, Gernot Heiser, Jeff Cotter and Adam Wiggins who have spent many hours discussing various aspects of the project. Also, David Johnson, who has given an enormous amount of time training students to solder, examining PCBs, and being generally helpful in a multitude of ways, must be acknowledged. He has also tolerated the Sunswift students' working in his office for several months.

Thanks must go to the U-Committee who provided the funds required to construct the system.

David Finn and James Kennedy from Tritium Pty Ltd have provided a huge amount of support in the implementation of software for their motor controller, as well as effecting repairs required following various incidents. The Biel school of engineering have provided tools and information necessary to develop their device further.

A large number of Sunswift team members contributed significantly to the project including (but not limited to) Monique Alfris, Lauren Tan, Bonne Eggleston, Kian Chin, Andrew Moran, Kushan Fernando, Owen The, Robyn Hyslop, Robert Reid, Luke Macpherson, Tim Wardrop and Andrew Pratley.

The work with the PLEB would not have been possible without the help and guidance of Harvey Tuch and Shane Stephens.

The members of the Vektor CAN mailing list provided a wealth of valuable information and advice regarding many of the issues discussed herein.

Lastly, thanks to Gernot Heiser, John Snowdon and Robert Reid for feedback on draft documents.

Contents

List of Figures

Chapter 1

Introduction

The sport of solar car racing was born in 1982 when Hans Tholstrup and Larry Perkins crossed Australia in the world's first solar powered car - the Quiet Achiever. Following the journey, Hans set up an event in order to promote the use of sustainable technologies and encourage research and development in the area. The *World Solar Challenge* (WSC) was first run in 1987 and is now a bi-annual event. Cars compete on a course which runs from Darwin to Adelaide, covering 3010km through the centre of Australia.

Sunswift II is the *University of New South Wales* (UNSW) entrant in the WSC and has competed in the 1999 and 2001 races placing 19th and 11th respectively. These positions were both considered by the team to be below expectation. In both circumstances, the car's electronics were partly to blame for the poor placing.

The car is a complex amalgamation of interacting electronic and mechanical systems. High efficiency power electronics control the solar panel voltage and in-wheel motor. The solar cells are embedded in an array which is also the car's aerodynamic form. Batteries are used as a nominal energy buffer.

Prior to the project outlined in this document, the solar car's electronic systems operated as independent units. A rudimentary telemetry system was used to monitor the car's operation and status and send information to a support car. The information available from this system was insufficient to make many necessary decisions within a race or event. Error reporting was non-existent, since the devices were not linked in any way to the telemetry system. Furthermore, the control system was extremely rudimentary. Perhaps the most significant problem was the inflexibility which the system afforded - adding new functionality often required redesign of the entire electrical system.

A flexible, powerful set of hardware and software infrastructure was required to allow the various electrical systems within the car to interact, providing telemetry, control, and management. This was provided through the use of *control area network* (CAN) technology. Hardware and software required to implement the system in the solar car were developed.

The system developed is a framework with which a variety of devices can be designed. Its trivial to extend, and could easily be used in an entirely different application should the need arise.

The infrastructure developed was used to implement an integrated telemetry and control system in Sunswift II. The system was constructed by students with the supervision of the author, and installed in the solar powered car. Several test drives were undertaken at various stages of the implementation.

Additionally, in order to accommodate the new framework, and to remove problems that were mostly to blame for the team's disappointing performance during the 2001 WSC, the operating software for the car's *maximum power point trackers* (MPPTs) was rewritten. The new software solves a number of serious problems with the device that hindered the car's performance.

This document is intended to serve both as a report on the implementation of the system, the MPPT software and the reasoning behind the various design decisions, and as a reference for the *solar racing team* (SRT) such that future work might be conducted, some suggestions for which are given.

Chapter 2

Background

2.1 Car Description

Sunswift II is a three-wheel solar-electric vehicle designed, built, and manufactured by the UNSW solar racing team. The project was commenced in 1995 as the fourth year thesis of Byron Kennedy, an Electrical Engineering undergraduate thesis student. In order to learn about solar powered cars, the team purchased Aurora Q1 from the Aurora Vehicle Association. The Q1 had competed in the 1993 WSC and placed 5th. UNSW raced the car, re-badged as Sunswift, and achieved 9th place. (For the remainder of this document, the name "Sunswift" will refer to the Sunswift II solar car. Where reference to the original car is necessary, the term "Sunswift I" will be used).

The design of Sunswift II was commenced following the 1996 WSC, with the intention of competing in the following race. The car was designed to be a high-performance solar vehicle: to win the race. As such it was designed to be as aerodynamic as possible, resulting in a highly curved shape. This has posed a number of manufacturing challenges (as outlined in a previous conference paper [42]), and gives rise to several issues involving the maximum power point trackers (discussed in Chapter 6). The car's aerodynamic performance is, as a result, very impressive.

The car consists of a chrome-moly steel spaceframe chassis with double wishbone suspension both front and rear. The driver's seat forms a structural part of the chassis. Steering is via the front two wheels. The outer shell, which also supports the car's solar array, is constructed from carbon fibre, kevlar, fibreglass and nomex composites. The bottom half of the outer shell is attached to the chassis, the top half being removable for access to the driver and car components. The "top half" will be referred to as the "top shell" or "array".

The car's wheels are student-designed and student-built. They consist of a carbon fibre-nomex composite sandwich. The tyres used are Michelin Radial solar car tyres which have an extremely low coefficient of rolling resistance. The carbon wheels are both lighter than the previously used aluminium wheels and have less windage due to the absence of spokes.

The array, as well as forming an aerodynamic shape, also supports the solar panels. These are constructed from some four thousand individual solar cells, encapsulated by the team using a novel technique to allow for the complicated curves, while still giving a finish resulting in low aerodynamic drag [42]. MPPTs are used to optimise the solar array's power output, acting as voltage converters such that the panel operates

Figure 2.1: Sunswift II driving into Adelaide at the end of the WSC

at its optimum point (see Chapter 6). These are mounted in the top shell such that only a single, two wire, power connection from the array is required. While subject to significant improvement, the array currently generates approximately 1150W in one sun conditions.

The car's drive system consists of a three-phase brushless DC motor and motor controller. The team owns several motors, but the most commonly used is the CSIRO in-wheel motor [39] due to its extremely high efficiency. The electronic motor controller performs the function of a commutator in a normal brushed DC motor and the team also owns a number of these devices, discussed in Chapter 7.

The car has a 36kg *lithium-ion* (Li-ION) battery pack consisting of 240 cells manufactured by Saft. The pack has a nominal voltage of 148V, a fully charged voltage of 164V, and a capacity of approximately 5kWh (giving a range of approximately 300km at 100km/h on batteries alone).

The telemetry system has undergone a number of significant changes through this project, yet a basic form exists: sensors are used to gather information about the car's operation. This information is transmitted to a support car, wirelessly, such that decisions can be made about the running of the vehicle.

A variety of miscellaneous electronics are also installed. In order to be deemed road-worthy, the car must have functioning indicators, brake lights, and a horn. In order that the driver has rear-vision, a small camera is mounted at the rear of the canopy. An LCD display presents the camera's signal.

Figure 2.2 shows the car's power usage vs. speed, given the values estimated for Sunswift II. Figure 2.1 shows the car driving into Adelaide following the 2001 World Solar Challenge.

2.2 Race Optimisation

Solar car racing is about efficiency. The car which uses the sun's energy to propel itself in the most efficient manner will, in the absence of exception, win the race. This involves two main strands of optimisation. First, the car itself must be mechanically and electrically efficient. The systems within the car must use as little energy as possible, the aerodynamic losses must be as small as possible and the car's rolling resistance must be minimised. The car must also be reliable; capable of surviving the arduous environment without failure, since only a very small time spent by the side of the road can affect the car's average speed.

The other part of solar racing is optimising the car's performance over a distance, for example, in a race. As well as being able to perform efficiently at any instant, it must also be possible to optimise the vehicle's overall performance.

Four factors in optimisation will be considered here: strategy, reliability, efficiency, and operation.

2.2.1 Strategy

Batteries are an integral part of solar racing. The battery pack of a solar car is used to buffer energy such that, in periods of high demand, energy is available to drive the car. In periods of low demand, the battery can be recharged using excess energy available.

The WSC racing period extends from 8:00AM to 5:00PM [2]. Before and after the racing period, the car, while not allowed to move more than a safe distance from the road, may charge the battery, tilting the solar array in order to maximise that charge. Additionally, the car must stop at seven control points along the 3010km course (at which the battery may also be charged).

The energy obtained via charging during these stops, and the variation in energy requirements along the course (e.g. due to hills) makes strategy vital. There exists an equation (equation 2.1) which will approximate the rate of energy usage of a solar powered car at a given speed [12, 40]. The resulting graph (Figure 2.2) makes it clear that running a non-optimal strategy (e.g. varying the speed erratically around an average) can be extremely costly in terms of overall energy usage and therefore possible average speed. As an example, were the car to travel at 100km/h for one hour, and 60km/h for another hour (giving an average speed of 80km/h), it would cover 160 km using 2.25 kWh. Were the car to have travelled at the average 80km/h the entire time, only 2kWh would be used. This is an extremely simplistic view and doesn't take into account a large number of important variables (e.g. environmental effects, hills...). For a more thorough discussion of solar car strategy, see Peter Pudney's PhD thesis [38], The Leading Edge *[44]* by Goro Tamai, and The Speed of Light by David Roche et al [40].

$$P = \frac{1}{\eta}\left[mgv\left(\sin\left(G\right) + C_{rr}\cos\left(G\right)\right) + \frac{1}{2}C_d A\rho\left(v - v_w\right)^3\right] \tag{2.1}$$

where:

m = mass (kg)

g = gravity (ms^{-2})

v = velocity (ms^{-1})

G = gradient (rads)

C_{rr} = Coefficient of Rolling Resistance

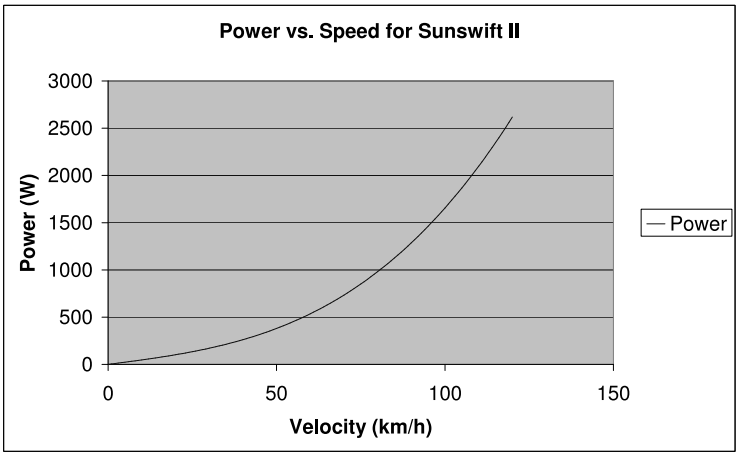


Figure 2.2: Power vs. Speed given by equation 2.1 using coefficients estimated for Sunswift II travelling on a level road.

CdA = drag area (m^2)
P = power used (W)
η = drive train efficiency (no units)
ρ = density of air (kg/m^3)

In order to make effective decisions regarding strategy, the persons responsible must be presented with adequate information. This information regards the weather forecast, the gradients and magnitude of the climbs and descents along the route to be travelled, and the way in which the car responds (in terms of its power production and usage) to these environments. The strategist must also be aware of the state of the car at every point during the race both so that its correct operation can be verified, and that the previously run strategy can be evaluated and future strategy updated.

While the weather and course profile are determined via the appropriate bureaus and pre-race surveys, the majority of the information is most effectively deduced using an appropriate telemetry system within the car. Various telemetry systems implemented by different teams have provided different functionalities (see Section 2.4), but the most basic system will monitor the current into or out of the array, battery and motor, the battery voltage, and the speed. These five parameters can give a reasonable idea of what is going on within the car. This is particularly when using Li-ION batteries which exhibit a strong correlation between the battery voltage and the state of charge (see Figure 9.3).

2.2.2 Reliability

Perhaps the most important of all qualities of the car to maintain and improve is its reliability. This cannot be stressed enough. Reliability is the key to successful solar-car racing**.**

To emphasise this point further take, as an example, the 2001 WSC, and the issues with the MPPTs observed during that race. The trackers arrived from Biel (the manufacturer) one month before the team was due to leave for Darwin. Some preliminary

testing was completed, essentially verifying that the devices seemed to perform in conjunction with a small test-panel and battery (which was not the race pack). The team assumed they would work as specified. The trackers were tested in-car, and operated without serious issue during the period prior to the race. On the start line, with a fully-charged battery pack, the trackers did not appear to function. It was assumed that the high battery voltage had caused the trackers to enter an over-voltage mode, and that they would function as soon as the pack voltage dipped slightly. As the start came and went, and the car began to drain the battery, the voltage began to drop. At 150V, the array began to generate power. This was over an hour and a half into the race. The pack had already been nearly half drained. Thus, approximately 2 kWh had been lost as a result. It was later found that the manufacturer had been in error when configuring the tracker's software.

A number of other issues were encountered during the 2001 WSC with regard to these devices, as outlined in Chapter 6. A particularly frustrating "feature" was the tendency to mis-track the maximum power point. Resetting the tracker's microcontroller, and thus the tracking algorithm, would often lead to a 300W increase in the solar-array power output. It is estimated that roughly 7kWh was lost over the duration of the race due to the various malfunctions of the MPPTs: equating to approximately 150W for the race's duration. Had this additional power been available it would have increased the car's average speed by approximately 7km/h placing the car among the top 10 positions (compared with the 11th place attained).

This is one example of a component which was unreliable and disadvantaged the team severely in terms of rank. There are many other examples, from other teams and races. Aurora, the 1996 event favourite, was forced to retire only minutes into the race when its brakes failed at a set of traffic lights on the way out of Darwin.

Other component failures in Sunswift included faulty circuitry associated with the motor controller, fairings rubbing causing excessive power usage and flat tyres, as well as problematic wireless communications.

Stopping the car should be avoided. If the car can keep moving, and can produce energy via the array, the decision is generally made to continue. The mathematics generally prohibits time by the side of the road. Ten minutes spent servicing a flat tyre would decrease the winning car's average speed in the 2001 WSC by 0.5km/h. An hour spent debugging an electrical fault could cost 3km/h. Note that at the speed travelled by Nuna, the winner of the 2001 WSC, 3km/h translates to approximately 100W (all calculations are based on Nuna's time in the 2001 WSC [3], using equation 2.1).

Having a component that does not function correctly can be extremely costly in terms of average speed, and having to stop in order to fix the problem exacerbates the issue. Again, with emphasis: **reliability is the key to successful solar racing**.

2.2.3 Efficiency

Perhaps obviously, another method of optimisation is that of improving the mechanical and electrical efficiency of the vehicle. Since the energy available is limited, the faster the car can travel at a given power, the higher the average speed is likely to be (given reliability and strategy are sufficiently strong). Also, the more efficiently the sun's energy can be converted to electricity, the faster the car will be able to travel.

Optimising the power used involves improving the rolling resistance, aerodynamic drag, and drive-train efficiency. Rolling resistance (corresponding to the C_{rr} term in equation 2.1) is the drag caused by the tyres on the road, bearing losses, suspension losses, etc. The power used is approximately linearly related to the speed of the vehicle.

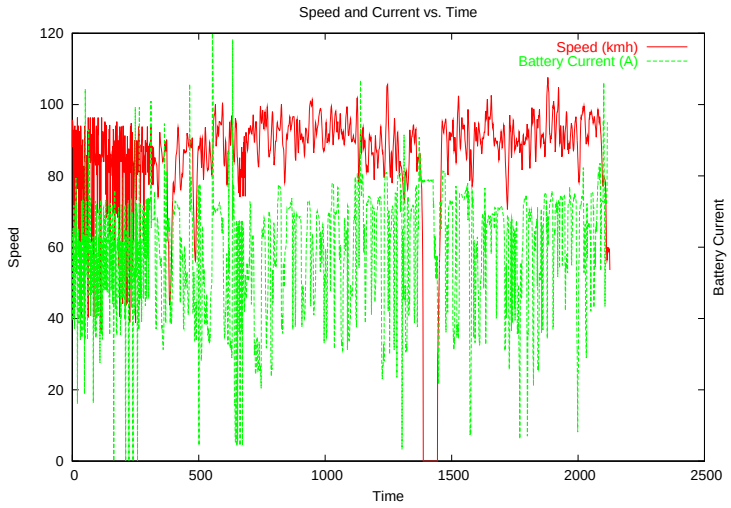


Figure 2.3: Current, Speed vs. Time for a testing run prior to WSC 2001

The aerodynamic drag is that which is caused by the car's motion through the air, and is affected by the amount of laminar, turbulent and separated flow, the frontal area, the amount of interference drag, etc. The power used is related to the cube of the vehicle's speed (and the CdA term in equation 2.1).

The drive train efficiency is the percentage of energy that leaves the battery that is expressed as work, moving the car. Losses involved in a car similar to Sunswift occur in the motor controller, motor, and any mechanical linkages involved (for example, a chain drive when using a motor not able to be mounted within the wheel).

Optimising the power produced involves obtaining (or building) the most efficient solar cells possible, using an appropriate encapsulation system, and ensuring the MPPTs operate correctly and efficiently. Reliability is again an issue, in that solar cell arrays are fragile and must be encapsulated such that the cells are sufficiently protected.

2.2.4 Operation and driving

The operation of the car is a final area which plays a large part in the success of a car during a race. Good drivers can have a significant impact on the overall efficiency and average speed. The motor controller has a non-linear efficiency vs. power curve, with a particular optimum. That optimum approximately coincides with the car's cruising power, but large efficiency drops significantly when the system is required to provide a large amount of power.

Non-optimal change to the throttle setting causes large power variation. As an example, two inexperienced drivers were involved with Sunswift II's 2001 campaign. Figure 2.3 shows current vs. time and speed vs. time graphs for a testing run prior to the race. The variation is obvious and caused a decrease in efficiency.

This can, in part, be blamed on poor feedback systems. A bike speedometer was the only instrument within the driver's field of vision. An analogue potentiometer with only a 300 degree rotation was used as the throttle, making it extremely sensitive. Also, it was difficult to provide the drivers with an evaluation of their performance: meaning that they were forced to guess as to how to improve.

2.3 Problems with Existing Telemetry/Control System

Sunswift II has never performed to expectation in the WSC, mostly due to poor strategy and unreliable systems, in particular, the electrical system, which consisted of several non-interacting components.

The 2001 telemetry and control system used a centralised topology. Analogue sensors were connected to a central data logger [9] via dedicated wires. The logger sampled once per five seconds.

The data produced were then transmitted digitally (via RS232) to a small embedded computer which collected it. Other sources of digital data could be connected to the same device via separate, dedicated interfaces (should they be available). Wireless Ethernet was used for communication with the support car.

Five critical channels were monitored, with other miscellaneous data (such as temperatures of major components) also being measured: vehicle speed, battery voltage, battery current, array current, motor current. These five channels provided a good indication of the status of the car, but left much to be desired.

A central control block acted as the driver interface unit. All driver controls connected to this central node, and all devices which required power at a low voltage (e.g. 12V) were supplied from this point using individual wires (again, adding to the wiring complexity and weight).

There were several issues:

- The operation and design of the components which failed during WSC were unknown to the team, and as such it was difficult for these components to be repaired during the race.
- There was no form of error reporting via the telemetry system.
- The sampling rate of the telemetry system was too low to make accurate battery-current integrations.
- Telemetry and control systems within the car were almost entirely analogue, which made them susceptible to noise.
- Long cable runs were unsightly, heavy, and complex. This limited the number of interfaces that could practically be used[1].
- Should the central nodes fail, the entire system also fails.
- The system was difficult to expand[2].

Essentially, functionality that could otherwise be utilised was wasted due to the inability of the car's systems to interact. The telemetry system couldn't access the motor controller's data or send commands to it. Further to this, the wiring within the car was extremely complex, which, in turn, led to a lack of flexibility. Any interaction between devices would have had to have been specified when the hardware was designed, or else inelegant interfacing solutions hacked together.

The situation was far from ideal. The ability of each device to be able to communicate with other devices was seen to be highly desireable. It would then be possible

[1] For example, it was unmanageable for the system to connect to each of the seven MPPTs' microcontrollers.

[2] The addition of further sources or outlets of/for data would have required hardware revision.

for a motor controller to provide a vehicle speed, and for that vehicle speed to be sent to a support car via a telemetry unit, as well as being displayed for the driver. A cruise control unit could also use the information to adjust the throttle setting. If a useful device, not originally considered, were to be developed, it would also have access to the vehicle speed. This allows for an evolving system where functionality could be added as it is required (and likewise removed).

These problems closely represent those found in industrial control and process systems prior to the introduction of embedded control networks. Large numbers of individual units are required to perform complex tasks. Those individual units are usually provided by a variety of manufacturers and have varied features and functionality. Large wiring looms were required in order for the units to communicate. Telemetry is (as it is in a solar car) desirable within this environment, and separate, dedicated systems were, required for this purpose.

A further representative problem is that found in traditional automobiles where large numbers of individual intelligent devices must interact. For example, the engine control computer may be required to communicate with the car's (computer controlled) gearbox to coordinate gear changes. The sound system may need to interact with the ignition systems such that when the engine is disabled, the speakers are also disabled to conserve the battery's charge. Power windows, headlights, and other devices traditionally require long cable runs to their respective controllers. Again, amounting to a large, complicated, wiring loom[3].

The approach taken in the Sunswift II, 2001, electrical system, is inherently limited due to the inability of the various intelligent systems within the car to interact. In terms of data gathering, information was lost since the power electronics and control system could not report to the telemetry system. As a result many parameters were measured more than once, and some interesting measurements were not taken at all. Similarly, because the implementation of more effective control systems (such as speed based cruise control) required time-consuming and error-prone hardware development (to modify the separate analogue electronics), they were modified only for design-error correction (thus demonstrating the difficulty with expansion).

A replacement system designed to resolve these issues — to be clean, flexible, expandable — was required.

2.4 Previous Solar Car Systems

Several teams have attempted to solve the problems outlined in Section 2.3 in a number of different ways.

2.4.1 Sunswift II, 2001 (non-interacting systems)

The Sunswift system is extremely relevant to this work, as the new framework has been used to replace it. The system is also useful as an example because of the similarity to a number of other teams' solutions (e.g. Aurora '99, Lake Tuggeranong College). See Section 2.3 for an overview of its construction.

[3] It has been estimated that the 50kg wiring loom present in some vehicles results in an 0.5L/100km drop in fuel economy [24].

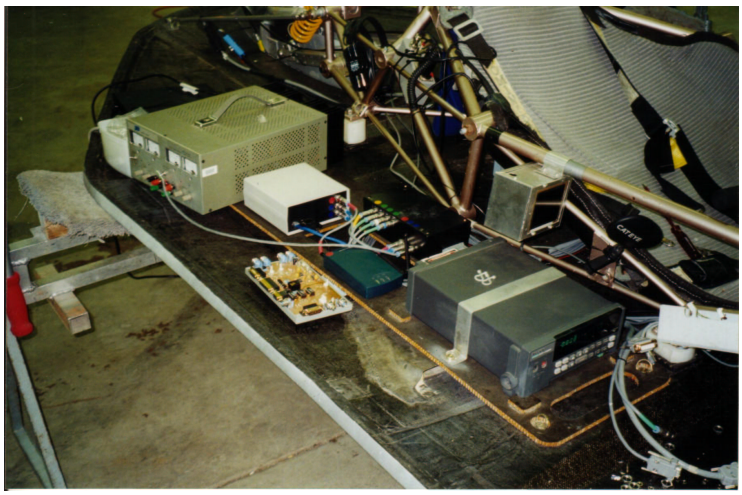


Figure 2.4: Sunswift II 2001 telemetry system

2.4.2 Aurora, 2001 (non-interacting systems)

Having a very similar system to the Sunswift vehicle, the Aurora solar car team employed the use of a MoTeC Advanced Dash Logger [31]. This logger gave a greater degree of functionality (over and above that provided by Sunswift's hydra [9]) through its large variety of interfaces, and driver display. By allowing the driver to view the operation of the solar car, it was better driven than those without such a comprehensive unit.

Controls in the Aurora car were also similar to those in Sunswift II; hard-wired with dedicated connections to the devices to be controlled. Modifications to the system are difficult, and so for each race, the system is almost completely rebuilt.

The Aurora team used trackers and a motor controller without available telemetry functionality (or a programmable control system); therefore this device was not integrated into the overall electrical system.

2.4.3 Iowa State University, 1999 (embedded network)

Iowa State University's 1999 team released much of the information related to their telemetry and control system into the public domain [46].

At its core, the system is essentially a number of nodes communicating via an RS485-based industrial field network. Above the RS485 network is a custom developed protocol called RDB. All devices in the car communicate using the RDB network.

A control "brain" was developed which was effectively a central, master node in the system. It connects to driver controls, as well as the motor controller and driver display via dedicated links (RS232). It monitors the driver controls and sends commands to the various devices (for example, the indicators, and motor controller). The car control brain was designed as the central hub in a star network (on the physical level). Each device connected to the car control brains.

The team developed MPPTs in house, and therefore had the opportunity to integrate RDB communications into the tracker itself. This allowed for a variety of information to be extracted efficiently, since the trackers need to make measurements in

the course of their operation. A variety of other devices also connected to the system. These included an amp-hour integrator, current monitoring equipment for the solar array, driver-display/controls, and the radio-modem interface.

The system is a good example of a network-based telemetry and control system. The standards involved are now open, and the system is well tested within a solar-car environment. The physical realisation of the network requires a star topology which increases cable run length over a bus topology.

Note that the system is inherently extendible via the addition of new nodes to the network; just as the addition of MPPTs to the network provides information regarding the array, so could addition of a motor-controller interface provide information regarding that device.

2.4.4 Sunshark, 1999 (integrated monolithic)

The *University of Queensland* (UQ) Sunshark car placed 3rd in the 1999 WSC and attained the "GM Sunraycer Technical Innovation Award" for the development of innovative electronics. Sunshark's strong performance was the result not only of an efficient car, but a reliable one — very little race time was spent making repairs.

The system was built specifically for the Sunshark car. Each component was built to interact in a specific manner with every other component. The number of components was reduced through integration.

The main components of the system were the motor controller, a driver display/controls unit, and a telemetry and control board. Dedicated connections were used to connect each of these components, as well as to link peripherals.

The motor controller was built specifically for the Sunshark car. Telemetry features were built into the controller. Communication between the telemetry board and the motor controller was via analogue signals and a *pulse width modulation* (PWM) encoded signal. The telemetry board required a number of sensors within the car; however, the links were short (due to the physical layout of the car), giving a clean result. Facilities were available to monitor digital data from a number of other sources (including MPPTs which were not team built, but had an interface available which was reverse-engineered).

The system provided a reliable method for control of the solar car, also giving the advantage of interaction between the components through integration. Each component of the system was dependent on each other component for operation of the whole. While the approach allowed a neat, operational environment, it has a number of disadvantages:

- Each component is dependent on each other component. It is not possible to run the motor controller without the operation of the telemetry board. Thus if one component fails, the whole system is likely to fail.

- It is difficult to add new functionality, or change functionality, without redesigning large portions of the system. For example, in order to add further temperature sensing to the motor controller it would be necessary to redesign both the motor controller and the telemetry board. In order to communicate with a different type of MPPT it would be necessary to redesign the entire telemetry board. Several features originally designed into the system did not function, and are now impossible to add without redesign/rebuild.

- A star topology is unsuitable for Sunswift II, where space for electronics is sparse. Since the space is limited, the electronics tend to be dispersed, leading to long wiring runs. A star topology requires significantly more wiring, in this circumstance, than a bus. Again, any change to the system is very likely to require significant wiring changes.

- The system is very specific and could not be used effectively in any situation other than within a solar-powered car or electric vehicle.

2.4.5 Solar Motions, 2001 (embedded network, non-interacting)

The Solar Motions team [1], following negative experiences with their MPPTs during the 1999 WSC, co-developed, with the Biel School of Engineering and Architecture, very high efficiency MPPTs for the 2001 race [35]. These devices became the basis for their telemetry system, measuring the input and output voltage and input current to each tracker.

The trackers were used as part of the telemetry system via a CAN network. The protocol layered above the CAN hardware is not well defined, but operated in a master-slave paradigm, ignoring the multi-master capabilities of CAN.

The interface to the CAN network was made via the same logic control board as that used in the power point trackers. This was connected to a radio modem which transmitted data from the solar car to the support car. A similar system was used by the Alpha-Centauri team [45] (who placed 1st).

While the Solar Motions team did use an embedded network, no attempt was made to allow systems to interact. The network was used purely as an information gathering device for a central node which then transmitted data to a support car.

2.4.6 Miscellaneous devices

Like the MPPTs designed by the university in Biel, other organisations have developed, and made available, either designs for solar car electronics, or manufacture suitable devices for sale. The Tritium *intelligent vehicle controller* (IVC) [29] is an example. These devices are capable of providing a wealth of useful information both before the race, to be used in debugging and tuning, and during the race, as telemetry and diagnostics.

These devices increasingly include hardware which can interface to a telemetry system. The emerging standard within solar car racing is the CAN specification, but there is no *higher level protocol* (HLP) specification such that the devices may communicate. Indeed, while these devices have been designed with the hardware necessary for communications via this bus, very little (if any) software had been written to take advantage of this.

2.4.7 General conclusions

Of the car's examined, the Sunshark system proved to be the most reliable. There are two reasons for this: first, the team had built almost every component of the car, and therefore could repair those components during the race. Second, a integrated system allowed for effective monitoring of the motor controller and MPPTs, as well as giving flexibility in terms of control.

While the ISU team's performance was lacklustre in comparison, their integrated electrical system provided similar flexibility. The approach taken allows easier expansion than that of the UQ team. The performance difference can be explained via malfunctioning MPPTs and a weaker mechanical design.

The solar motions and Aurora systems provide little more than data gathering services, since it is composed of several non-interacting systems.

By integrating the various systems within the car, far more information regarding its operation can be ascertained. An embedded network appears to offer the most flexible solution for integrated telemetry and control. By abstracting and standardising at the network level, modules can be changed and reconfigured in order to meet new needs and developments.

Whatever features the system provides, it must be reliable — the car should never be required to stop due to an electrical fault.

Chapter 3

Design

The UNSW SRT intends to actively develop Sunswift II to improve its performance. The car's previous electrical system provided an operational system, but not one that could be improved without rebuild. One goal of this thesis was to develop only a small part of the car's electronics: a framework with which components of the overall, integrated, system can be developed.

3.1 Requirements

A set of requirements were developed to resolve many of the issues outlined in Section 2.3 — a new electrical system for Sunswift II should:

- allow the collection of data from a variety of different sensors and transducers. The properties to be sensed may be located at any location in the car's 4.5x2x1m enclosing parallelpiped. These properties might include currents, voltages, frequencies, temperatures, or any other factors of the car's operation. Sensors may be placed in the solar array, which disconnects and is removed from the main body of the car;
- allow the collection of data gathered in the operation of devices within the car. An example of such a device is the MPPT which perform voltage conversions between a solar panel's optimum operating voltage and the car's power bus voltage. The power point trackers measure voltages and currents in order to determine the maximum operating point of the panel. This information should be made available to an operator. These devices may be located anywhere within the car's volume;
- provide status information on a device's functioning, and mode of function. (e.g. MPPT tracking mode). Diagnostics should be possible. (e.g. IV curve sweeps of the solar panels);
- provide feedback to the car's driver;
- perform well in an electrically noisy environment. Noisy supply lines, and electromagnetic interference should form an acceptable operating environment;
- perform well in a mechanically harsh environment. Sustained vibrations, dust and dirt, sharp shocks, should all be well tolerated;

- provide facilities for control of devices within the car. (e.g. the motor controller's set speed);
- be quickly re-configurable and repairable. Should a hardware fault require a component replacement, that component should be quickly removable and the new component require little to no configuration (in order to minimise the time off the road). The number of spare components should be minimised;
- be fault tolerant. One component's malfunction should be isolated to the failure of that component. The rest of the system should remain unaffected. In the event that large parts of the system do fail, any data that is collected by remaining parts of the system should be recoverable once the system resumes operation. No single failure should require the car be stopped for intervention;
- use a minimum of energy. A Figure of less than 10W is acceptable, less than 5W being preferable;
- be physically small and lightweight. Any cabling required should be minimised;
- have minimum cost and be within the solar car team's modest budget.

3.2 Macro-Design

Since the previous system was required to be completely rebuilt in order to make useful modifications, an entirely different paradigm could be employed. The car's previous system consisted of entirely non-interacting modules. While this provides some reliability improvement since there is essentially no interdependence of module upon module, the advantages possible using an integrated system such as that of Sunshark or PrISUm are obvious:

- more information is available, for less complexity
- efficiency can be improved since fewer independent measurements need to be made
- complex control solutions can be implemented without the worry of over-complicated interfacing
- significantly reduced wiring

The last two are the primary reasons the automobile industry has begun to use embedded control networks [24] in place of traditional wiring looms. This application has, in fact, been a driving force in the development of the technology (leading to the development of CAN [16]).

To give an integrated system within Sunswift, two major components are required to be interfaced; the MPPTs, and the motor controller. The rest of the telemetry/control system really serves as a slave to the needs of these two devices, since these are required for the essential functionality of energy generation and consumption. The rest of the electrical system must provide:

- driver controls and feedback;
- indicators, brake lights, horn, rear-vision;

- telemetry;
- other miscellaneous functions (e.g. control of battery cooling fans).

As has been established, developing the 2001 system to provide the desired functionality would have been virtually impossible. The MPPTs and motor controller both provide a digital interface. Hence, rebuilding the system, standardising on digital interfaces, results in a cleaner, neater solution.

Several types of digital communication have been used in previous solar cars (as outlined in Section 2.4). While a new style of data-transport system could have been employed (e.g. Bluetooth, infra-red, etc.), this would be beyond what was achievable in the course of this project and would provide little extra benefit. Furthermore, the use of tried technology allows more guarantees about its reliability.

An embedded network provides a flexible, power efficient, cost-effective paradigm to work within, as a standard way to transfer digital data between devices within the car.

3.3 Topology

A centralised system had been used by the Sunswift team for telemetry, but had a number of significant limitations as outlined in Section 2.4.1. In order to address the issues with the centralised topology, a bus topology may be employed. In this paradigm, instead of communicating with a common node, the devices communicate with a common bus. The bus forms a communications link between all devices.

Advantages of this distributed topology are:

- All interfaces are standardised. Each device must be able to communicate with the bus via its defined protocol, therefore only one interface/protocol must be implemented.
- The bus is expandable. When new devices are required, they may be added to the bus.
- If one device becomes faulty, the entire bus need not be disabled.
- Analogue signals can be sampled very close to the source, reducing the effects of EMI.
- Each device communicates digitally, making it easier to add galvanic isolation.
- Products with digital interfaces are easy to interface to the bus (e.g. MPPTs, motor controller).

Disadvantages associated with a bus are:

- Extra hardware is necessary to communicate with the bus at each device, increasing cost and complexity at the device level. Each device is required to communicate digitally.
- The system is more distributed, creating synchronisation and timing issues (for example, time stamps may not be synchronised).

There are very few other topologies that are significantly different to the bus topology at the physical level (excepting wireless communications). Ring networks are an example (as used in ControlNet [5]).

The advantages of a bus topology over a centralised data-gathering topology were found to be worth the initial extra complexity. It was determined that solutions to the problems outlined were available, and practical.

3.4 Bus Selection

A large number of digital bus technologies are available. Many of these are designed and specified for control. Others have been adapted to the application.

3.4.1 Bus requirements

The following requirements were placed on the network technology selected:

1. Sampling Rate — The network must be able to support the transmission of 10 samples/second from up to 100 sources/channels. The ideal technology would be capable of supporting higher sample rates and a larger number of channels. Up to 25 nodes may be present on the bus. Imagining a protocol that requires eight bytes per sample (time stamp + value + error correction), this equates to 8000 bytes/sec $\approx$ 8 kB/sec actual throughput.

2. Hierarchy — It is possible that sub-busses may exist for communication with specific devices, or where the main bus is over-powered. It is assumed that these will require a bridging device of some description.

3. Real-time — The system may have real-time requirements. The bus may form part of a control loop, or be used for emergency messages.

4. Transmission Range — Nodes will be placed along a cable length of up to 25m.

5. Noise — EMI will be a concern in the highly noisy environment within the solar car caused by the power electronics' switching.

6. Power — The system should use less than 1W (as a guide) to communicate via the bus, including any isolation and voltage conversion losses.

3.4.2 CAN

CAN [16] was developed in the late 1980s for use in the automotive industry. In this capacity it is used as an interface between the various digital components within a car such as engine management and electronic gearboxes: time-critical tasks requiring a real-time communications channel. Because of its simplicity, and the ability to prioritise packets, it may also be used to connect meters, switches, power windows, central locking, and other functions.

CAN is a serial protocol requiring two wires; a balanced differential pair. It satisfies the lowest two layers in the OSI (open systems interconnection) model [48]. CAN exhibits relatively high bit rates, good EMI rejection, and effective error detection and correction protocols. The protocol is optimised for short messages of between 0 and 8 bytes, with either an 11-bit or 29-bit identifier. Collisions are non-destructive with the

higher priority message being transmitted, leading to nearly 100% available bus utilisation. Errors and collisions are detected - the packet automatically being resent. The highest transmission rate possible is 1 Mbps (assuming a bus length of less than 100m). Many microcontroller devices are available with inbuilt CAN controllers. Stand alone controllers are also available. These can be interfaced to a microcontroller.

CAN has since been used in the manufacturing and aerospace industries. It forms the lowest levels of DeviceNet [5], CANOpen [21], and CANKingdom [14]; three common industrial networking protocols, as well as other higher level protocols.

All power electronic devices currently used in the solar car can communicate via CAN 2.0B (Biel MPPTs, and Tritium motor controller).

Some advantages are:

- high data transfer rates (1MBps);
- optimisation for small messages (for example, for transferring telemetry values and commands);
- data link layer is implemented in the controller;
- real-time capability, prioritised messages;
- excellent noise immunity with error detection/correction in hardware;
- simple to provide galvanic isolation;
- all power devices in the solar car are already CAN 2.0B capable;
- hardware detects and corrects errors.

Disadvantages include:

- controllers and transceivers are required;
- large amount of complexity at each node.

3.4.3 RS-485

RS-485 was developed to solve the transmission issues present with RS-232 communications. RS-485 defines the physical layer of the OSI standard: requiring a data-link layer, and higher layers to be implemented.

The standard allows an engineer to implement small, inexpensive, bidirectional master-slave and multi-master networks. Up to 32 nodes may be present on the network (as specified by the standard) but some line-driver manufacturers provide ICs with far lower than unit load, allowing more than the specified number of nodes. RS-485 performs well in noisy environments because it uses balanced differential signals over twisted pair wires. Further shielding of the cable is possible to reduce EMI further.

RS-485 is easy to implement. A microcontroller's inbuilt UART is often used to produce signals which are then fed into an RS-485 line driver to be connected to the network.

RS-485 is currently used within the car to communicate between the driver control board and the motor controller. All major devices have a UART available for communications, although do not necessarily have provision for the RS-485 line drivers.

Some advantages are:

- high transfer rates;
- the UART required is a built-in peripheral on most microcontrollers.

Disadvantages include:

- transceivers are required at each node;
- the data-link layer is left to an HLP, increasing software size and complexity;
- collisions are destructive.

3.4.4 Ethernet

Ethernet [43] was developed at the Xerox *Palo Alto Research Center* (PARC) in the 1970s by Dr Robert M. Metcalfe. It uses a *carrier sense multiple access / carrier detect* (CSMA/CD) scheme for its *medium access control* (MAC) mechanism.

Ethernet is a widely accepted specification of the data link layer, used for a variety of networking needs. Industrial protocols have been developed over Ethernet and TCP/IP. Ethernet has a high data transfer rate (up to 1000Mbits/sec).

Ethernet is traditionally non-deterministic, meaning that, particularly under high loads, the network will demonstrate variable packet latency. It was developed for the office environment and therefore has not been hardened against industrial environments[1]. Higher layer protocols are required to implement network, transport, and application layers. This is typically TCP/IP (even in industrial networks using protocols such as Ethernet/IP).

The use of Ethernet (and appropriate protocols) allows easy integration with the Internet and wireless networking devices which have been developed for both the industrial and office/home markets.

Data delivery is not guaranteed. Upper level protocols must be capable of detecting and re-sending dropped frames.

Frames accommodate a destination address as well as a data field. The data field can vary in size between 46 and 1500 bytes.

Wireless Ethernet is currently used as the point-to-point link between the solar car and support vehicle. Proxim RangeLAN2 [37] transceivers are used for communication between the solar car and support car.

Some advantages are:

- high data throughput;
- variable frame size;
- well adopted networking standard;
- isolation is built into the physical-layer specification.

Disadvantages include:

- complex controllers/hardware;
- optimisation for office networking;

[1] RJ-45 connectors, as used by 10Base-T, 100Base-T and 1000Base-T Ethernet, have been found to fail in the conditions within solar cars by many teams; unless appropriately reinforced

- no real-time capability;
- complex higher-layer protocols;
- repeater hardware required for some topologies;
- unreliable connectors specified;
- high power usage.

3.4.5 I^2C

I^2C [41] was designed by Philips Semiconductors in the early 1980s as a simple, serial, synchronous, bidirectional, 2-wire bus for *inter integrated-circuit* (I^2C) communications. The bus was originally developed for tightly integrated electronic systems to reduce wiring and system complexity. Devices interfaced might include EEPROMs, data converters, and other peripheral chips for microcontrollers. The standard is being actively developed, with the most recent specification released in January, 2000.

A large number of integrated circuits are available with I^2C built in. Many microcontrollers are able to communicate using the standard, and when hardware facilities are not available in these devices software can be used to emulate the bus. Many integrated circuits that would traditionally interface via a parallel bus to a microcontroller, are available with an I^2C interface; including ADCs, DACs, EEPROMs, etc.

$I^2C$is a true multi-master bus specifying collision detection and arbitration behaviours. Two bus lines are required, a serial data line, and a serial clock line. Both of these are bidirectional.

The bus is designed to transport single-byte messages, and can transfer up to 100kbit/sec in standard mode, and 400kbit/sec in fast mode. It uses conventional TTL/CMOS logic levels as the physical layer.

I^2C is not currently used within the solar car.

Some advantages are:

- many devices have a built-in I^2C interface;
- no transceiver/controller required.

Disadvantages include:

- not designed for communication over long wires;
- comparatively low data rate;
- if a microcontroller is not used, programmability/flexibility at the nodes is limited.

3.4.6 1-Wire

1-Wire [36] has been developed by Dallas Semiconductors (now Maxim Integrated Circuits) as a minimalistic inter-integrated-circuit bus. Two wires are required, ground and a single communications/power wire. Power can be provided parasitically over the data line[2]. Integrated circuits can therefore be made with only two links to the host controller or microprocessor.

[2]Because the devices use very little power, they can draw small amounts of current from the communications line while it is in the high state. This eliminates the need for a separate power wire.

Because of these features, 1-Wire has been adapted to be used for low-cost sensor networks. Temperature sensors and EEPROMs can be packaged into small surface-mount devices with very small outlines. The very small number of wires means the packages only require two pins for ground, power and data communications. A total of two-wires (ground and 1-Wire communications) can supply power and communications to a large number of sensors. The 1-Wire specification was originally intended for local communications, but information is available regarding the network parameters for communications over distances greater than 10m.

Devices operate under a master-slave regime — a centralised topology. The master sends requests to sensors, and those sensors respond with their actions.

1-Wire can transfer up to 115kbit/sec in overdrive mode and 9600bit/sec in standard mode; however, all addressing occurs in standard mode. Because of protocol overheads (such as device discovery, etc.) actual bus throughput is far less than either of these might suggest.

1-Wire has been considered in the past for use in implementing large sensor arrays, but is not currently used in any form in the solar car.

Some advantages are:

- very small, very low cost devices are available with 1-Wire interfaces;
- two wires can carry both data and power.

Disadvantages include:

- very low data rate;
- the sole manufacturer is Maxim Integrated Circuits;
- licence limitations are imposed on slave implementations;
- single-master: centralised logical topology.

3.4.7 Comparison table

Table 9.1 gives information current as of January, 2002. All prices are the manufacturer's if possible. If not, Digikey Corporation was used as a standard vendor. HLP refers to features which are/can be implemented in higher layer protocols.

3.4.8 Conclusion

While there are advantages to the simplicity of systems based around the I^2C, and 1-Wire buses, as well as advantages to the high bandwidth available over Ethernet, the choice was made between CAN and RS485.

I^2C was eliminated as not providing the bus length and noise immunity required. 1-Wire does not provide the data rate required for the sophisticated data gathering system envisioned in the requirements. Also, the inability to create slaves limits its use as a general purpose data transfer bus within the car. The power saving due to each of these technologies', compared to the more powerful RS485 and CAN networks, was found to be insignificant . Ethernet does not have the real-time or soft-real-time capability of the other busses. Also, in its 10Base-T incarnation, it requires a central hub.

	CAN	RS485	Ethernet	I^2C	1-Wire
Physical Properties					
Max #Nodes	Driver dependent 110 (UC5350)	32	1024 (10Base-T)	Max 400pF (Typically 40)	~100
Max Length	40m - 6km (Data rate limited)	1200m	100m (Cable Dependent) (10Base-T)	8m	~200m
Max Data Rate (Mb/s)	1	10	10 (10Base-T)	0.1 (Standard I2C)	0.01 (Standard)
# Wires	2	2	4 (10Base-T)	3	2 (Includes power)
Transmission Mode	Differential	Differential	Differential	CMOS/TTL	CMOS/TTL
Termination	Yes - 120Ω	Yes - 120Ω	NA	No	No
Topology	Bus	Bus	Star	Bus	Flexible
Duplex	Half	Half	Full	Half	Half
Built-In Devices	Yes	Yes	No	Yes	Yes
Message size (Bytes)	8	NA	48-1500	1	NA
Cable	Twisted pair	Twisted pair	Twisted pair	NA	NA
Connector	NA	NA	RJ-45 (10Base-T)	NA	NA
Cost/Availability					
Manufacturer	Many	Many	Many	Many	Maxim/Dalsemi
Controller Cost (US$)	2.28(@1k) (MCP2510)	2.79 (@1k) (MAX3100)	8.80(@10k) (CS8900A)	NA	$2.06(@1k) (DS2480B) (1 per bus)
Transceiver Cost	2.17(@1k) (SN65HVD231)	1.75 (@1k) (MAX3485)	(Incorporated into controller CS8900A)	NA	NA
Noise Immunity					
	Good	Good	Good	Poor	OK
Shielding	Twisted pair wiring	Twisted pair wiring	Twisted pair wiring	NA	NA
Signal type	Differential	Differential	Differential		NA
Error Detection	CRC	Parity check/HLP	CRC	Acknowledge	CRC
Error Correction	Automatic resend	HLP	HLP	HLP	HLP
Real Time Capable	Non-destructive collisions Prioritised messages	HLP	No (can be made quasi-deterministic)	HLP	HLP
Power Req.					
Ext. Controller	10mA@5.5V (MCP2510)	150μA@3.3V (MAX3100)	45mA@3.3V (CS8900A)	NA	NA
Ext. Controller (Sleep)	5μA@5.5V (MCP2510)	10μA@3.3V (MAX3100)	1mA@3.3V (CS8900A)	NA	NA
Transceiver	17mA@3.3V (SN65HVD232)	1mA@3.3V (MAX3485)	NA	NA	NA
Transceiver (Sleep)	1μA@3.3V (SN65HVD231)	NA	NA	NA	NA
Higher Level Protocols	DeviceNet (ODVA) SDS (Honeywell) CANOpen (CiA) CANKingdom (Kvaser)	DataHighway Profibus	TCP/IP Ethernet/IP Profibus on TCP Modbus/TCP	ACCESS	MicroLAN
Complexity	Medium	Medium	High	Low	Low
Intended Purpose	Automotive inter-μP communication	Industrial control	Office networking	Inter integrated circuit bus	Low power/cost Inter-IC bus

Table 3.2: Comparison of various networking technologies

CAN and RS485 are both in common use in industrial networks. These networks are used in conditions which are very similar those within the solar car. CAN was originally designed as an automobile control and data-transfer system. As such, it is very well suited to the solar car environment. While greater complexity is required at each node for both of these busses in the form of controllers, transceivers, and protocols, the advantage is that each node is able to process data in a true multi-master network.

CAN was chosen for the following reasons:

- each device required to be connected in the solar car (Biel MPPTs and Tritium motor controller) is already equipped with hardware to communicate via the CAN bus;
- CAN is partially real-time capable;
- more of the upper level protocol is generally implemented in hardware for CAN, requiring less intervention from user-designed software;
- CAN is already optimised for short packets, such as those intended within the car;
- CAN has been proven over a number of years in the automotive industry.

3.5 Higher-Level Protocols

A large number of communications protocols (one web site claims over 40) exist for use as transport and application layers over CAN (selected in Section 3.4.8 as the network technology for the framework being developed). These are, for the most part, industrial control networks, although many form flexible communications frameworks similar to that required for Sunswift II. Some companies (e.g. HMS [32]) have implemented hardware that complies with many of the different standards.

Functionality provided by higher layer protocols in this context include:

- start up behaviour
- message ID specification and distribution
- status reporting
- data fragmentation[3]
- baud rate specification
- configuration
- global clock synchronisation

Several options were considered, and are outlined in the following sections. Some other protocols include J1939, NMEA2000, SDS. These essentially provide the same functionality as CANOpen and DeviceNet, and therefore were not reviewed. For a good comparison between three high level protocols, see Lernartson and Fredriksson's discussion of SDS, DeviceNet and CanKingdom [25].

[3] CAN only transports 8 data-bytes per packet.

3.5.1 DeviceNet

DeviceNet [5] was developed as a low cost solution to the problem of industrial control network's hard wiring between devices such as limit switches, indicators, etc. Large wiring looms are condensed to the two-wire serial bus making them more manageable and faster/easier to install and maintain. The technology also allows for bridges to EtherNET/IP, allowing a TCP/IP based network to interact with the control network. Management systems can therefore communicate with automation systems allowing for analysis of processes, and early failure warning.

DeviceNet is intended to put intelligence in most devices in an industrial situation. It is the most widely used of the industrial control network standards, and as such is the best supported, with the largest number of suppliers of compatible devices.

The protocol allows for a range of transfer modes useful in a solar car environment. DeviceNet is based on a producer/consumer model. The protocol is highly bandwidth-conscious, with a number of features allowing for the reduction of traffic (e.g. transmit-on-change). Packet fragmentation techniques are built into the protocol to allow for messages larger than the eight bytes allowed for by the CAN specification [16]. Peer-to-peer transfers are specified, as well as master-slave communications.

Advantages of using the DeviceNet protocol are:

- DeviceNet is a common standard with industry support. Many devices are available and can inter-operate with devices from different manufacturers in a multi-vendor system.
- DeviceNet's protocol features fit very well within the solar car environment.
- The protocol has been tested and proven in environments similar to that within a solar car as well as those with more strict reliability constraints. The risk involved with a standard such as DeviceNet is far smaller than that associated with a home-grown protocol.

Disadvantages are seen to be:

- While the protocol can be cut down and small versions of the code required have been used in microcontrollers with limited resources, DeviceNet has a large footprint requiring the use of higher-powered microcontrollers than otherwise necessary.
- The specification is not free. The price of the DeviceNet protocol specification is a significant cost for Sunswift, other solar car teams, and other groups wishing to use the resulting framework.

3.5.2 CANOpen

CANOpen [21] is intended as a multi purpose transport protocol, layered over CAN. It provides facilities useful for a number of applications. It features automatic network configuration, cyclic and event-driven data exchange behaviour, and the possibility of determinism.

The protocol is maintained by the *CAN in Automation Users and Manufacturers Group* (CiA) [19]. In addition to automation systems, the CANOpen standard is used in cars and ships. It has also been used in medical equipment.

CANOpen has the following advantages over other solutions with respect to a solar car application:

- CANOpen is a widely adopted standard for communications over a CAN bus. Inter-operability between the solar car's system and off-the-shelf units is a possibility.
- The protocol is well tested and revised.

The following are seen as disadvantages associated with CANOpen:

- The latest specification is not free, and is too expensive to be purchased by the team (however an older version is freely available [21]).
- CANOpen is a large protocol and would necessitate a more powerful microcontroller than is otherwise necessary.

3.5.3 CanKingdom

CanKingdom attempts to solve the problem that many protocols (including CANOpen) implement services for nodes. This is a non-optimal fit for CAN due to the broadcast nature of the protocol. In a CAN system, the situation is naturally reversed — nodes serve the network. For example, a traditional connection-based network would require two "hosts" to form a connection. Alternately, one host might send a message to another host (using an address to specify the *destination)*. A CAN network does not lend itself to this style of protocol, being a broadcast network where every node can receive every messages (or a subset of the messages filtered by their identifiers in hardware). These lower layers inherently support multicasting. A protocol which can only support point-to-point connections ignores this capability.

CANOpen implements the OSI model [13], and as such suffers from the above symptom. OSI is designed to connect two clients in a network so that they can share information. In order for this to occur, each node in a CANOpen system needs substantial knowledge about the other nodes in the system.

CanKingdom, as mentioned, attempts to solve this issue. The nodes serve the network. The nodes are not required to have prior knowledge about the network. It does, however, require that a master (a "capital" in CanKingdom terminology) be present for configuration of the nodes. A capital has complete knowledge of the system and coordinates all activities.

CanKingdom has the advantage of being an open standard. It can also accommodate CANOpen, DeviceNet, J1939 and SDS devices (as well as devices from other protocols). The protocol takes far less program and memory space than the previously mentioned standards.

Advantages:

- CanKingdom is smaller (in terms of RAM/ROM footprint) than most other standardised higher layer protocols.
- The protocol specification is freely available.
- Difficult problems such as clock synchronisation have already been considered.
- Supports extended CAN.

Disadvantages:

- The network is dependent on one node — the “capital”. This central, controlling node, inherently decreases the reliability of the system.

- CanKingdom provides a large number of features which are unlikely to be used in the framework under construction.

3.5.4 Time triggered CAN

Time triggered CAN (TT-CAN) is a higher layer protocol that is supported by some CAN hardware. It is not discussed in depth here, but is summarised well by Thomas Fuhrer et al [15]. Time triggered CAN has the advantage of being a hard real-time network (as is required for brake-by-wire, steer-by-wire, etc). Since real-time performance was not essential to this application, the protocol was not considered.

3.5.5 Home-grown protocol

There are a number of arguments for designing/implementing a custom protocol:

- If commercially available modules are not likely to be used, the standardisation is not useful in terms of interaction with externally developed devices.

- A custom-built protocol can provide exactly the features required, without extra overhead.

- Some standard protocols are extremely large. For example, an efficiently implemented stack, the Kvaser CANOpen implementation, uses 16kB ROM, 6kB RAM [23]. For comparison, the microcontroller eventually used has 32kB ROM and 4kB RAM.

- The time involved with understanding and implementing an available standard is significant.

- Any additions, or extensions, to the protocol can be made without compatibility issues.

- Using a minimal set of functionality means that the protocol can use more of the hardware features available. Many protocols such as CANOpen do not use CAN features such as remote frame requests, and hardware message filters.

A custom-built protocol can provide almost all the functionality that a standardised protocol can because the solutions used in open standards can be duplicated. Configuration methods can be implemented to suit the application, and the size of the protocol scaled to fit in the intended hardware.

There are disadvantages. Well established standards have been scrutinised by more proficient eyes than any member of the solar car team, and therefore are more likely to be reliable. If a reference implementation could be obtained, it’s possible that the time for implementation could be greatly reduced compared with a home-grown protocol.

3.5.6 Conclusions

Having studied the available options, the decision was made to write a custom protocol. This decision was made for the following reasons:

- Most of the available protocols (barring CanKingdom) require a more elaborate processor than is required for the actual functionality implemented at each node. If a large protocol had been chosen, the hardware would have been more expensive than otherwise. The MPPT uses a microcontroller with only 8kB ROM. To use CANOpen (for example) would eliminate the possibility of interfacing this device. By writing a protocol in-house, the complexity of the protocol can match that of the system.
- Various features from the different available protocols can be integrated into a small, tightly-integrated code-base.
- Use of a larger proportion of available hardware features can provide for a more efficient implementation (since we wish to be able to implement on a low performance microcontroller, this is important).
- The system presently only needs a very simple protocol, but should this change in the future, the protocol could be expanded.
- A large number of other projects (including many car manufacturers) use custom protocols for the above reasons.
- The protocol can, in this case, be made so simple that the design is not difficult and that implementation time is minimised.
- Reliability can be built into the design of the protocol.

The development of this protocol is studied further in Chapter 5.

Chapter 4

Hardware Infrastructure Development

The goal of this project was to develop a set of hardware and software infrastructure for the solar car. Hardware infrastructure, in this context and as a product, refers to the design of a physical base on which a variety of projects can be built. This design takes the form of a reference design for a CAN network node. The reference design is expressed as a set of schematics, and board layouts. These designs were eventually built (as is outlined in Chapter 8) and tested.

A number of critical design decisions were made in the development of Sunswift's hardware infrastructure. This chapter outlines a number of these decisions, and describes the products that resulted.

4.1 Overall Car Design

It is intended that all electronics within the car be based on the hardware infrastructure to be outlined here. The various functions should be implemented via the use of various nodes connected to a network. The bus should also carry power for the nodes, to reduce wiring.

As an example, the indicators might be implemented using a node which receives commands as to whether to turn them on or off (the design of such nodes is the subject of Chapter 8). Likewise, the horn might use a similar device on the network which turns a siren on and off. A current sensor might be implemented using a node which sends CAN packets at various intervals (see Chapter 5). Furthermore, any electronic device will be connected to the CAN bus, since the same physical cable will provide power to low-voltage devices (which will normally be able to communicate via the data lines). This significantly reduces the amount of wiring required.

Through the integration of every component into the same system, the distinction between telemetry and control is blurred. Data is transferred on the bus with varying priorities (since CAN allows for priority levels [16]).

With the above in mind, a reference design was developed.

4.2 General Principles

Some general principles were adhered to during the design:

- Where possible, components available from Farnell [4] have been used. While the component may be less costly from a different source, and that source may be used as a supplier when time permits, Farnell's same-day delivery is extremely useful when a particular part is required at short notice (e.g. two days before a race).
- Where low-power design becomes an obstacle for a more important design goal (such as reliability), it is likely that the power consumption should take second place (within reason).

4.3 Bus Design

The first specification to be made was that of the physical connection between nodes.

It was decided to daisy-chain the nodes together by having two connectors at each node. This allows for the easy addition and removal of the nodes without significant re-wiring of the rest of the bus. One disadvantage is that part of the bus will lose power when it is unplugged (in order to add or remove a node at that point). The bus will, in this case, also lose the termination resistor at one end of the data lines, so more errors will occur on the part of the bus that is connected. This should not prove to be a problem since most nodes are essentially stateless and should be re-enabled successfully when re-connected. If this proves to be a problem, more than one power supply can be added to the bus[1]. This design can be contrasted with that used by the MPPTs which use a single connector with screw-terminals. In order to modify the wiring loom, the entire loom needs to be rebuilt.

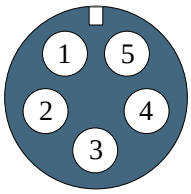
The function of each pin (the pin-out) on the connector was determined to some degree by a previous low-voltage specification. A low-voltage power bus with a similar daisy-chaining philosophy had been used in the previous low voltage system [47]. Despite the entire system being rebuilt, the pin-out was carried over (with the addition of a 5V power connection). The connector used also carried over: a five pin, male (on the cable) connector from the Triad range by Thomas and Betts. These are rated to 3A (for the 5 way version), 60V, have good shielding properties, and are relatively small. A PCB mount version of the connector is available eliminating unreliable "fly leads" which are required to connect boards to panel-mount connectors. They are screw-locking and IP-65 rated (which includes a sealing specification). These were used in the 2001 system, and proved to be extremely reliable with no failures due to faulty connections (apart from those due to operator error — e.g. not screwed in). The disadvantage of such high-quality connectors is the cost: over $2000 was spent during this project on connectors alone.

The pin-out is given in Table 4.1.

Two power rails are used. The 5V rail is used to supply the logic circuits. The 12V rail is used to supply power for devices. These voltages were chosen to minimise conversion, and resistive losses. It is intended that 3.3V is used to supply logic at each node, and that a linear regulator be used to generate that voltage from the 5V line. This

[1] In Sunswift, for example, a DC-DC converter providing power could be situated at each end of the physical bus such that if a break occurs, all nodes will remain powered. Two breaks would have to occur in one conductor to disable any node.

Pin	Function
1	5V
2	12V
3	GND
4	CAN High
5	CAN Low



(a) Solder side

Table 4.1: Pin out of the bus connectors specified

arrangement means that noise on the power line (at 5V) should be filtered out by the linear regulator at the node, as well as minimising the effects of voltage drop across the (long) power supply lines. The separate supply rail increases the efficiency by allowing an efficient switching converter to generate 5V from the 12V line (at a power source), as opposed to a linear regulator at each node from 12V (since switching regulators have a substantial zero-load current, having one at each node decreases the efficiency when compared with a linear regulator).

Twisted-pair cable is necessary for the differential pair (the CAN wires) to EMI rejection. Finding cable that can handle the connector's current rating of 3A (ie carry power), as well as having the twisted pairs (for the communications signals), is difficult. Certain types of category 5 cable (as used for 10Base-T Ethernet) are appropriate, and this has been used in the initial implementation and should support signalling at CAN's maximum specified speed of 1MBps.

4.4 Isolation and Protection

One obvious concern when connecting every device in the car to the same network, is that if one device fails, the entire network may fail with it. Take, for example, the situation where an exposed thermocouple in the battery comes into contact with a terminal at some (high) voltage. If the battery pack ground were connected to the ground in the CAN bus, it is likely the node would experience a large current through some path or network of paths. As a result it is likely that it would be damaged. Consider the situation where the path includes one of the CAN bus wires. In this situation the entire car's electronic system could experience the high voltage. This situation is unacceptable due to reliability, maintainability and cost concerns.

Several protective measures can be taken. The most commonly used to protect against over and reverse voltage is to place a zener diode across any inputs to be protected. This clamps the voltage to the zener's reverse breakdown voltage. Some zener diodes are manufactured particularly for this task (e.g. the Transorb range of transient voltage suppressors). If a fuse is placed in series with the connection to be protected, the fuse will open-circuit if an over-voltage condition occurs. This solution, while simple, has some issues when used to protect the CAN data lines. The capacitance added by the transient suppressors can make the bus behave improperly. In order to protect the node as much as possible from damage, the power rails are protected in this fashion (see Figure 4.1).

One method of further protecting the bus is to ensure each node is galvanically isolated. This means that no electrical connection exists between each side of an isola-

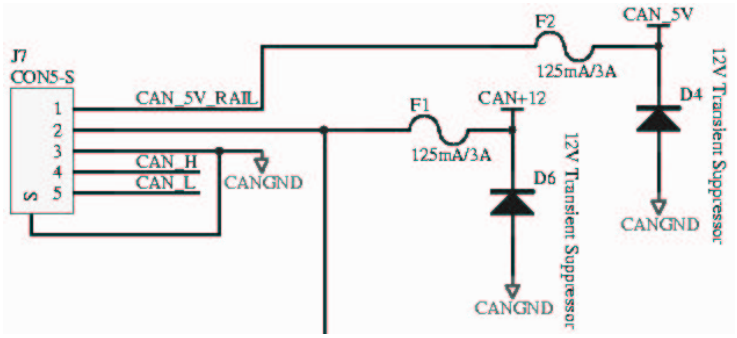


Figure 4.1: Transient suppressors protect the power rails from over-voltage.

tion barrier. Digital signals may be coupled using optocouplers (where an LED shines across a physical barrier forming a logical connection, but not an electrical one), and power may be coupled using magnetic fields (as in a transformer). This kind of isolation protects in both directions. The device is protected from mis-operation of the bus, and the bus is protected from mis-operation of the device.

Perhaps most importantly, devices which do not require the CAN bus to operate — e.g. the MPPTs and motor controller — will be able to continue operation even if the entire network has been damaged, since they are on the opposite side of a physical barrier. Furthermore, since each node is floating relative to ground, a short can occur to the battery positive, and the node will simply float to that potential and continue operation. It is only when a second short *on the same node* occurs that damage will occur. This need for *at least* two errors before the system can be damaged is a compelling reason for the use of isolation.

The disadvantages associated with this kind of protection are cost, power usage and propagation delay[2]. High quality isolators are expensive devices. The quality is generally inversely proportional to the power usage and propagation delay. Propagation delay is important in a CAN system because each node must be able to assert the bus, have the signal travel through the optocoupler, through the transceiver, to one end of the bus, and back again back through the transceiver and isolation to the controller, in one bit time. The speed of light limits the length of a CAN bus for this reason. If significant propagation delays are introduced because of the optocouplers either the bus' baud rate (which is related to the bit-time), or the bus length, will be limited. Isolated DC-DC converters (required to isolate the power supply) are also expensive.

The decision was made to galvanically isolate all nodes in the system, as well as keep the CAN ground separate from the battery ground. This means that at least two errors are required to cause damage. See Figure 4.2.

Isoloop isolators, which are magnetic, rather than optical, isolators, operate at the 3.3V required and have a propagation delay of only 5ns. These devices have very small power consumption when compared with traditional optocouplers [33].

Small isolated DC-DC converter modules from Newport can be used to provide an isolated DC source. These components provide galvanic isolation to over 1000V and are specified in the reference schematic/design. These isolation components form a substantial proportion of the total cost

This step has already proven its worth, protecting the majority of the system

[2]Propagation delay is the amount of time taken for the signal to travel through the optocoupler.

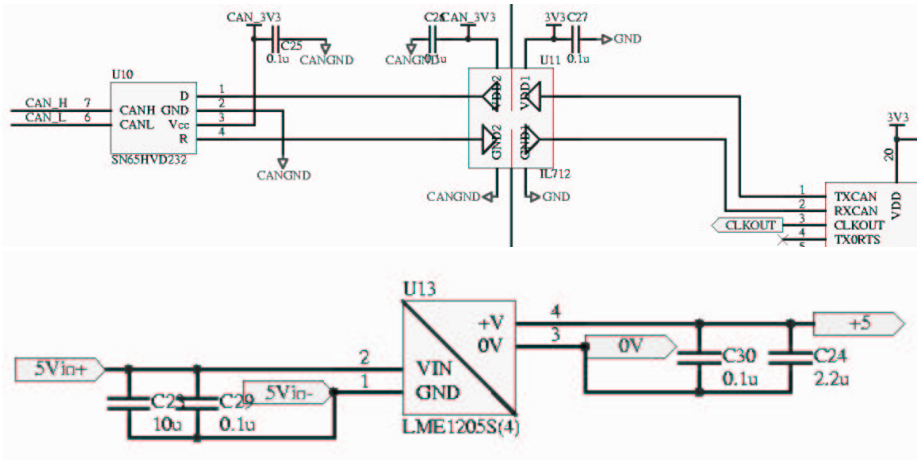


Figure 4.2: Schematic of isolation barriers.

through two malfunctions. In one case, a metallic object came into contact with the DC-DC[3] input rails, vapourising a large part of the input circuitry, but leaving the network unaffected because of the isolation barrier which was present. In another case, a motor controller failure was isolated to the motor controller itself.

4.5 Redundant Busses

Redundant CAN busses are sometimes used to give greater reliability in applications such as satellites and safety-critical control systems.

In a typical [24] automotive use of CAN, there will be two networks. One, typically running at a low rate, will accommodate non-critical control tasks, and user comfort tasks. This non-critical network will also attempt to use as little power as possible via the use of ”sleep” modes. A second, high-speed, network is be used for real-time traffic such as engine management, anti-lock brakes, cruise control, etc. By having physically separate prioritised networks, the reliability, and real-time performance, of the high-speed network is significantly increased.

Satellite networks [28] (and other susceptible, or particularly critical applications) often use redundant busses [26] such that in the case of a bus failure, the system will continue to operate using a second (or third) bus. The concept is equivalent to the use of redundant microprocessors in highly critical systems. Various CAN controllers are available with multiple interfaces [20] and would suit this purpose.

The decision was made to use a single bus for the following reasons:

- cost and complexity — the team can not afford to increase the cost of the system more than absolutely necessary;
- incompatibility — neither the Biel trackers, nor the Tritium motor controller support multiple CAN busses;
- power consumption.

[3] A device which converts from one voltage to another, and is used in Sunswift II to generate appropriate supply voltages from that of the battery.

Thus the probability of total bus failure was deemed to be small enough that its disadvantages make a redundant bus undesirable. The performance of the system could, however, be improved by using a prioritised bus separation similar to that outlined for typical automobiles. This does not bear heavily upon the standard node design since in either design only one CAN interface is required.

4.6 Standard features

The following features were specified as being standard on any node within the network:

- power indicator LED (green) — two indicator LEDs (one red, one yellow) for use in fault-finding;
- ambient temperature sense — the motor controller and trackers both have temperature sensing, which can provide important diagnostic information. For very little added complexity every node can be given this ability;
- sleep capability — each node should be able to put itself into a low-power mode;
- power usage measurement - each node should be able to (at least) estimate its own power consumption;
- real-time clock - each node must be able to tell the time since boot at millisecond accuracy.

4.7 Miscellaneous Design Decisions

The following further decisions were made:

- Protel was chosen as the schematic capture and *printed circuit board* (PCB) layout tool due to its easy to use interface, and the author's relative knowledge of the package. An excellent alternative is provided in the CADSoft's Eagle. Two clear advantages of Eagle are its ability to run under the Linux operating system, and its free evaluation version. The *School of Electrical Engineering and Telecommunications* (EE&T) at UNSW owns a number of licenses for Protel, to which the SRT has access. The platform dependence can be forgiven for its more productive interface.
- *Surface mount technology* (SMT) has been embraced in these designs since it provides more reliable, smaller PCBs. The *School of Computer Science and Engineering* (CSE) provided the team with the facilities to produce very detailed boards. The 0603 package size was standardised upon such that the team might invest in volume orders of commonly used discrete parts.
- PCB layout was done according to manufacturing specifications of the various producers used during the course of this project. This includes CSE's prototyping facilities which were used extensively, and proved to be an invaluable tool in quickly refining the designs. Two layer (or single layer) designs were used when possible due to the large costs associated with multi-layer boards.

- All designs have taken into account housing and mounting considerations.
- 3.3V was chosen as the standard logic voltage due to reduced IC power consumption compared with 5V.

4.8 Reference Design

A reference design was developed such that future network-node development might be speeded up. This reference design consists of a set of schematics and a PCB layout for a particular device.

The schematics are extremely modular. A hierarchical design means that modules can be reused very easily, in the same way functions in a software library can be reused. The interfaces are relatively obvious. During the implementation of the Sunswift system (as outlined in Chapter 8) several boards were designed; these schematics proved invaluable both to the author, and to the team members involved.

In the development of the reference design, a large number of schematic components and component footprints were drawn. These form a comprehensive library which members of the team have since used to develop their own designs. This library forms the core of the hardware infrastructure and should speed any future development.

The “CANRefNode”, as it is named, consists of a main board with two CAN connectors, and two “daughtercard” connectors. The daughtercard connectors allow a personality board to stack on top of the CANRefNode, to provide the overall device with input, output, and other functionality. This approach was taken such that a large number of CANRefNodes could be manufactured and tested. Low-cost personality boards can then be developed in order to build and modify the system. The CANRefNode, provides a node designer with a working base which can be customised using the daughtercards. It is assumed that the Sunswift system will evolve over time and it is hoped that the CANRefNodes will rarely require replacement, affording a substantial saving in both development time and cost. A reference design for the daughtercard has also been produced.

The CANRefNode provides the basic functions expected of each node in the network (as outlined in the previous sections).

4.8.1 Microcontroller

A number of different solutions were examined with regard to a processor/CAN-controller combination.

Initially, an integrated processor/CAN-controller was sought, since this would reduce the number of chips, as well as the software complexity. It was soon realised that using such a device was likely to require a more powerful (and power hungry) microcontroller than desired. Since the functions likely to be carried out by each node are very simple, the microcontroller also need only be simple. Some vendors have released CAN controllers with in-built digital and analogue I/O such that a microcontroller is not required. Since this board was intended to be flexible, and multi-purposed, this option was not considered. At the time of selection, the Microchip PIC18F series, with integrated CAN controller, had not been released. Its use may provide a superior solution.

CSE had considerable experience with the AVR range of devices. They are extremely good, 8-bit, general purpose, high-performance RISC microcontrollers. A

number of useful peripherals are available built in to various models. Furthermore, excellent software development tools are freely available in the form of avr-gcc, simulavr, uisp, etc. An in-system programmer can be manufactured at little cost (as outlined in an application note [7]), but this was not necessary since the team had purchased an STK500 AVR development board for a previous project. CSE had recently purchased 50 ATMega323s [8] and this particular device was therefore immediately available. Unfortunately, no AVR has an integrated CAN peripheral, so using the devices in this context requires the use of an external controller. Due to these advantages, and the author's previous experience with the devices, the ATMega323 was selected for use in this design. Note that, as of September, 2002, the ATMega323 is not recommended for new designs. The ATMega32 is a more recent replacement which is able run at higher clock rates.

Two options, not available at the time of design, which may prove more appropriate for future evolutions include:

- the MSP430 by Texas instruments, which appears to be an excellent alternative, being a 16-bit processor with better peripheral options, using 1/16th the power for a comparable model;
- the PIC18FX series of microcontrollers from Microchip. These devices integrate the MCP2510 CAN controller with an 8-bit microprocessor providing a small, low power CAN node with very few external ICs.

An external crystal of 3.6864 MHz was used for the microcontroller in order to allow the built-in UART to clock at normal PC serial port rates with little error.

4.8.2 CAN Controller

The choice of the AVR as a microcontroller immediately necessitates the use of an external CAN controller. This decision is perhaps the most regretted of those made in the hardware design. The MCP2510 CAN controller is manufactured by Microchip as a lightweight device to connect small embedded systems to a CAN network. It offers three transmission buffers and two receive buffers. It interfaces via a *serial peripheral interface* (SPI) bus (which the ATMega323 implements in hardware).

Again, very few other controllers were considered because of the MCP2510's obvious advantages:

- it was designed for application in low performance systems and so uses less power than alternatives;
- bandwidth limitations placed by the use of the SPI interface should not be an issue since the nodes should not require high volume traffic (although the network should be able to handle the throughput);
- the Biel MPPT uses the device meaning that a substantial amount of code reuse could be made if the same controller were used.

Another suitable controller is the SJA1000, which is bus-interfaced (as opposed to the three-wire SPI interface used by the MCP2510). This controller is slightly more powerful and power-hungry than the MCP2510. The SJA1000 is more mature than the MCP2510, having been developed by Philips, a company with a much longer history in CAN device manufacture.

Since no significant complaints had been discussed with the MPPT developers and other CAN developers, and it had the obvious advantage of code and experience reuse, the MCP2510 was selected as the CAN controller. With perfect hindsight, this decision might not have been optimal.

The MCP2510 has an extensive errata document [30] detailing numerous silicon bugs. This document had been read at the time of the device's selection. The effects were thought to be non-critical; however, later they caused and continue to cause significant problems (including packet loss and garbled messages). At least one further silicon bug has been discovered through the use of the device during this thesis.

16.0MHz was chosen for the CAN crystal to allow the controller to attain its top speed of 1MBps (were it not prohibited by a silicon bug). While this rate is not required in the present implementation, the design allows the use of this high rate if required in the future.

4.8.3 Power supplies and isolation

An un-isolated and an isolated power supply are required on-board. An un-isolated supply is required to provide power to the CAN line driver (an SN65HVD232 from Texas instruments), and one side of the logic isolators at 3.3V using the CAN bus' 5V rail as a source. This is realised via a *low drop-out* (LDO) linear regulator (a switching regulator was calculated to be less efficient due to the low load current).

An isolated supply is required to provide all isolated circuitry. The isolation barrier was placed between the CAN controller and line driver so that only two logic isolators are required (an IL712 from the Isoloop range of isolators provides both in one package). A 250mW isolated converter from Newport (LME0505S) provides 5V to the microcontroller side of the isolation barrier using the CAN bus' 5V rail as an input. This is then filtered and regulated using an LDO linear regulator.

The current out of the converter is measured using an 0.82Ω resistor and a MAX4173 high side current sense amplifier connected to the microcontroller's *analogue to digital converter* (ADC) providing an instantaneous current measurement of the 5V line on the microcontroller side of the isolation barrier.

4.8.4 Miscellaneous hardware

The following miscellaneous hardware was added to the design:

- 31-pin Conan connectors from Molex were used to stack the daughtercard upon the CANRefNode.
- Three LEDs are used to indicate 3.3V power, and give two general purpose indicators from the microcontroller.
- An LM50 temperature transducer connected to the ADC provides an ambient temperature measurement.
- A 32kHz watch crystal was connected across the timer clock pins to be used as a real-time clock.

4.8.5 PCB layout

The CANRefNode was laid out using a two layer board and through hole plating. Components were placed on both sides of the board in order to reduce size.

A Dick Smith Electronics "Zippy" box was chosen as a low-cost plastic enclosure. The PCB was sized such that it mounted via the inbuilt PCB rails. Four M2.5 mounting holes were provided both to secure the daughtercard (using 6mm hexagonal plastic standoffs) as well as provide an alternate mounting method, should the box not be used.

A physical isolation barrier was designed into the PCB, with the node and CAN sides of the isolation barrier being physically separated (all circuitry directly connected to the CAN network is situated at one end of the board).

Usual design principles were adhered to with clock traces being kept as short as possible, ground planes being used to provide a good reference, etc.

All unused pins of the microcontroller are made available to the daughtercard (via the Conan connectors), along with power supplies from both sides of the isolation barrier. One pin is designated as a "Daughtercard shutdown" pin. Upon assertion, the daughtercard is required to use as little power as possible. All pins, excepting the CAN-side power lines, are on the microcontroller side of the isolation barrier. The isolated power supply can provide up to 30mA at 3.3V, which is enough for most logic circuits.

Two revisions to the PCB were made. These were prototyped using an LPKF ProtoMat C60, made available by David Johnson at CSE. The boards were through-hole plated using a copper plating system commissioned via the first board produced (some time was spent assisting with this). The rapid prototyping facilities saved a large amount of money in wasted PCB manufacture, and the final product quality was significantly increased via the ability to make these revisions.

A daughtercard reference schematic/PCB was also designed. In order to develop a daughtercard, the original design can be copied, and components added in order to provide the function necessary.

4.9 Results

As is outlined in Chapter 8, 20 CANRefNode PCBs were manufactured by IMP Printed Circuits. This manufacturer was chosen for these PCBs because of their low cost, high definition service — required because of the detail of the PCB.

The reference schematics were also used as the basis for several boards (again, outlined in Chapter 8) successfully. The existence of these designs and modularity with which they are constructed makes implementing a new node trivial.

The reference design for daughtercards was used to construct seven different personality boards. The time for development for a CAN node with individual function and purpose thus reduced to under an hour's design and layout (for a simple daughtercard). See Figure 4.4 as an example of a daughtercard (a current sensor) connected to a prototype CANRefNode, mounted in the intended box. Note that the connectors were positioned such that all cables are in-line (current is measured by putting a cable through the hole in the large black transducer).

Code sufficient to test the operation of the various pieces of hardware was written using CodeVisionAVR studio's evaluation demo. This provided for rapid development on the AVR processors with comprehensive libraries built in.

Schematics and PCB artwork are given in Appendix A.

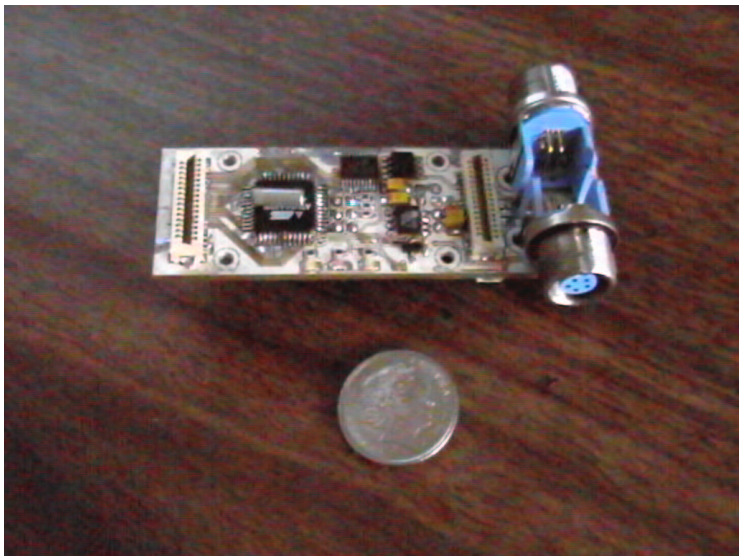


Figure 4.3: Prototype CANRefNode (with 5c piece for scale)

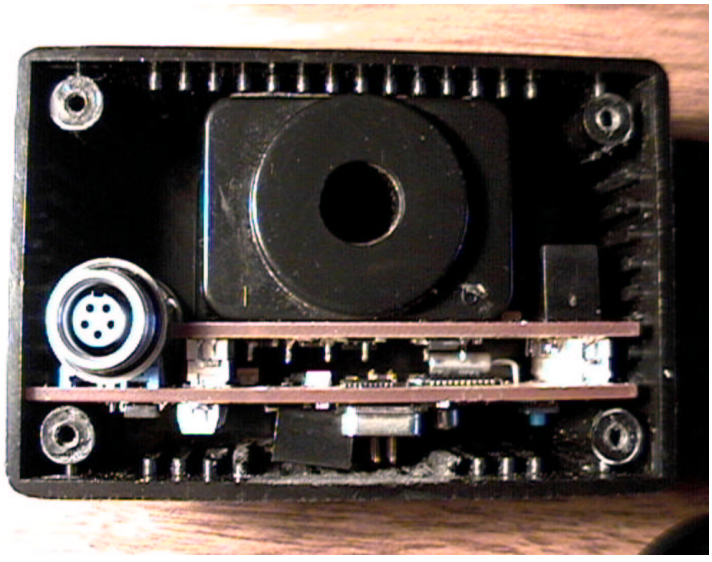


Figure 4.4: Current sensor prototype mounted in box

4.10 Externally Specified Systems

Since the Biel MPPTs and Tritium motor controller are externally specified, only feedback or modification is possible. These systems were modified, and add-in boards manufactured in order to integrate them into the system (as outlined in Chapters 6 and 7). It was found that while it was impossible to implement the above specification entirely, it was possible to sufficiently emulate it. Hopefully future revisions of this hardware will include the remaining features.

4.11 PLEB

The UNSW/CSE PLEB [49] was intended to be used as a processing node, performing any functions which the low power, low performance nodes would have difficulty with. Examples include the wireless communication, which was to be implemented via Sunswift's RangeLAN 2 access points from Proxim. These require an Ethernet interface. The PLEB is a StrongARM SA1100 based computer. Both ARM Linux, and research operating systems based on the L4 micro-kernel [27] have been successfully booted and used in applications on this platform.

The StrongARM processor runs at 200Mhz, and can have up to 64MB RAM, 8MB flash. A number of internal peripherals are available. The PLEB's stacking daughtercard arrangement (similar to that used for the CANRefNodes) is used for expansion. Ethernet and IDE daughtercards have been developed and proven.

An MCP2510-based daughtercard was developed. Unfortunately, it was built and never tested, owing to its dependence on an integrated Ethernet/IDE board which had been designed and constructed by CSE. The Ethernet/IDE board was manufactured, but Linux drivers were never completed (and are, at the time of writing, still under construction). The author's knowledge of L4 was not sufficient to allow the construction of a solution in a reasonable time-frame.

Another extremely useful daughtercard is also currently under construction by CSE students. It interfaces a compact flash card. This would allow booting from, and logging to, flash storage, or an IBM microdrive.

It is recommended that the PLEB CAN board be redesigned with a more powerful, memory-mapped CAN controller, with readily available Linux drivers. The MCP2510 is very much geared toward a small, low-power system, accepting the limitations of these (such as low data-transfer rate over SPI, small number of message buffers, etc). The PLEB does not have such limitations, and would benefit from the use of a more powerful CAN controller such as the SJA1000. This device has drivers for Linux available under GPL.

In order to interface the PLEB to the network (without the CAN daughtercard mentioned above), one of its serial ports was connected to a CANRefNode carrying a serial port daughtercard. A small protocol was written to exchange data between the two, acting as a CAN-to-serial bridge.

Chapter 5

Software Infrastructure Development

Given the hardware platform/specification outlined in Chapter 4, the CANRefNode boards which resulted, the Biel MPPT, the Tritium motor controller, and the PLEB, also required was operating software to allow the nodes to perform their functions. Furthermore it was necessary to develop communications software to allow the various parts of the system to integrate via the CAN network.

The above platforms have different architectures. The Biel MPPT uses a Microchip PIC microcontroller, the Tritium motor controller uses a Texas Instruments DSP, the PLEB uses a StrongARM running Linux, and the reference boards specified in Chapter 4 use an Atmel AVR.

Building a software infrastructure became a matter of building a set of libraries and drivers such that the above systems could operate, and inter-operate. Furthermore, the libraries are portable, with platform-specific code isolated to individual modules.

A requirement was that the environment which the infrastructure provides be easy to use, and learn. It is intended that students with very little programming experience be able to program the boards. A second requirement was that the infrastructure, and the CAN protocol in particular, be extendible such that should new requirements be determined, they should be implementable.

5.1 Compilers and Tools

Various tools were used in the construction of the various pieces of software to be outlined here.

It was decided to use C as the language of choice (using assembly where required) for programming the various embedded systems for portability, speed of development, and ease of use reasons. Careful choice of compiler was necessary, since code size and speed often limit usefulness, and therefore the better the compiler optimisation, the more the overall system will be capable of. Furthermore, microcontroller C compilers, in general, have a number of oddities, which may limit code compatibility. There are a large number of compilers being developed for the architectures and devices of the processors used.

AVR-GCC was chosen as the compiler for the AVR because of its GPL license, associated libraries, and ANSI C support. The compiler is under active development.

It has been compared favourably to several commercially available compilers (worth several thousands of dollars each on the market) by developers on the AVR-GCC user's mailing list. A CodeVisionAVR evaluation version was used for some tasks because of its comprehensive set of libraries, and code template generator. In evaluation, the performance of AVR-GCC has been very good. It provides good headers and libraries, and optimises reasonably well. A version running under Windows was used because of good integration with the IDE and programmer interface tools available from Atmel. During the period of this project, the limitations associated with the Linux version have ceased to exist because several tools necessary for effective development on Unix based platforms have matured significantly.

CCS PCM was used to compile for the PIC 16F877 which is used by the Biel MPPT because the Biel team had invested in it, and code compatibility was important. The SRT paid for a cut down version of the full compiler. It uses a large number of C extensions. It also places several limitation, such as a lack of function pointers, in order to allow performance and size optimisations. This compiler has a number of serious limitations, including lack of support for function pointers.

CodeComposer from Texas Instruments was used as the compiler for the TMS320L2407 used in the Tritium motor controller for similar reasons to those above.

Prior to this project, the team's revision control system was purely manual. This led to a number of occasions where code was lost, out-of-date code was used, etc. In order to resolve these problems, *concurrent versioning system* (CVS) was set up and used. All code and documentation resides in a CVS repository on `solarch.sunswift.unsw.edu.au`.

Debugging software running on a system with very few feedback mechanisms (as is the case with many of these devices) is difficult. A Tektronix TDS210 two-channel *digital storage oscilloscope* (DSO) was used to speed development, making measurements of both analogue and digital inputs and outputs. The storage and measurements features both proved invaluable. This tool made debugging and verifying the various systems far more precise. It was often the case that a single measurement would save hours of debugging time.

5.2 Abstraction Layer

Because the system has been developed on four platforms, code which can be portable will save a large amount of development time: particularly in the case of drivers for devices such as the MCP2510 CAN controller, which take a significant time to perfect.

Hardware abstraction is most commonly achieved through the use of an *operating system* (OS), which provides a generic interface to applications as well as a number of useful functions and constructions. A real-time operating system can speed the development of applications with real-time requirements (which many embedded systems have). Several real-time OSs (or more correctly, executives, since they are so limited) are available for the platforms discussed, but no OS, or abstraction layer, is available across all the platforms.

Because the applications intended for the majority of nodes were intended to be extremely simple, a single-platform operating system provides little benefit. Real-time requirements can be satisfied through direct use of interrupts and timers. Because of this it was decided not to specify any particular operating system, and build an abstraction layer library on which applications can be developed.

OSs are, however, used in some nodes. In this case, the abstraction layer library is built on top of the OS's existing interfaces (e.g. in Linux, which runs on the PLEB).

An abstraction layer for CAN hardware already exists in the form of CANpie [22]. This API specification provides an interface to the various features of CAN controllers and is freely available. Some implementations of CANOpen and other HLPs have used CANPie. Various drivers also provide an interface for operating systems. These layers were not used because it entails a large code size which could not be afforded. A very small interface was developed through the initial development of a driver for the MCP2510 (which in turn, uses an SPI driver abstraction such that it is portable).

An interface to peripherals such as a *universal asynchronous receiver transmitter* (UART) (if present), a *real-time clock* (RTC) of some kind, *electrically-eraseable programmable read only memory* (EEPROM), etc. are specified, and implementations of those interfaces developed for all platforms currently in use.

Despite a large number of different targets, and the absence of an OS on all but one, most code, except for the abstraction library, is compilable across all the platforms. This has speeded the development of difficult components such as the MCP2510 driver, which is used on the PIC, AVR and PLEB platforms. A bug fix on one platform easily applies to the others. The same applies to the protocol library. The Tritium motor controller CAN communications code was developed in less than three days through the development of a set of abstraction libraries for the TMS320L2407.

5.3 Protocol

A simple protocol, known as "Scandal" was developed, based on a library linked with the abstraction library for each platform as outlined above.

5.3.1 Requirements and design

The requirements for the protocol are extremely simple. It must provide:

- a way of sending telemetry over the network
- a way for other nodes to sensibly use that information
- configuration facilities
- a way of reporting status and errors at each node

Network reliability features need not be built in because the hardware already performs sufficient error-checking and retransmission.

CAN is organised as a consumer/producer style network, and it makes sense to take advantage of this (as is outlined in the CAN Kingdom protocol specification [14]). Thus a similar "nodes serve the network" philosophy was taken with the design. The network is considered an entity from which information is sent and received.

State is stored in the persistent storage (often EEPROM) which is required of all platforms.

Time is given by a millisecond-accuracy timer, which is also required. It is intended that the protocol synchronise these clocks. This has not yet been implemented.

The protocol is based around the concept of channels. A channel represents a stream of numbers. The channel value may be updated regularly or irregularly, and therefore the source may be event — or time — triggered. The source of a channel

provides data to the network. The receptor of a channel receives information from that source. A channel can have multiple receptors, but only one source (mainly because CAN prohibits the transmission of identical identifiers from different nodes). The term "out-channel" is used to refer to the channel source, the term "in-channel" to the destination.

An example to illustrate: a current sensor, monitoring the battery current of the car, has an out-channel representing the value of the current. It is updated at 10Hz. A driver display has several in-channels which correspond to the various values displayed on the screen. The driver display in-channel is configured to listen to the out-channel on the current sensor (as identified by its node ID and channel ID). The display software updates the value on the display whenever the value of the in-channel is changed (via network activity).

Another example: a set of indicators is used on the car. Three nodes control six lights (front left and right, rear left and right, and rear brake lights). Each node has two in-channels which control the state of two lights. Whenever the in-channel is updated, the state of the light is updated. If a non-zero value is received, the light is on, otherwise it is off. A driver controls board controls the indicators, and has two out-channels representing the state of the left and right indicators respectively. A positive value is sent when the indicators should be on, and a zero is sent when off. Thus the four in-channels corresponding to the indicators can be associated with the two out-channels from the driver controls.

Note that either signals or events can be represented via these values, and thus a separate message type is not required.

Heartbeats are used to inform the network of the operation of each node, as well as some values regarding its state.

Errors are reported explicitly.

Extended identifiers (29 bit) are used for all messages such that a large number of new message types can be added in the future.

Every node has an address which is stored in the persistent memory. Each node is initially configured with an address of zero, and is reconfigured over the network. The nodes are connected to the network one by one such that an initial address conflict does not occur, and their addresses can be updated. A protocol to determine a temporary address at run-time may be implemented in the future to avoid this initial configuration stage.

Since all nodes are configured prior to use, and store their configuration internally, the failure of any node will not mean the entire network will fail.

Device details are defined in *scandal_devices.h*.

A polled architecture was used to simplify synchronisation. The handle_scandal function must be called as often as possible (generally once through the main loop), and no less than 10Hz. This function handles in-channels, heartbeat messages, resets and configuration. It also calls can_poll, which may or may not have an effect, depending on the particular CAN abstraction layer being used.

5.3.2 Message types and identifiers

Every CAN message has an identifier, which can either be 11 bits (standard) or 29bits (extended). Extended identifiers were chosen to simplify the protocol, and to allow for expandability.

In Scandal, every message identifier starts with two fields: priority, and message type ID. The priority is placed in the *most significant bit* (MSB) of the identifier. This

Type#	Type	Description
0	Channel	Update a channel
1	Configuration	Update a configuration parameter
2	Heartbeat	Sent periodically to announce operation to the network
3	Error	Report an error
4	Reset	Reset a particular node

Table 5.1: Message type IDs

3: Priority	8: Message Type	8: Source Addr	10: Channel Number

Table 5.2: Channel message identifier

means that higher priority messages will win arbitration on the bus, since they will have dominant bits earlier in the transmission (e.g. Scandal priority becomes CAN priority). This is important, since CAN performs non-destructive collisions, with the highest priority message being transmitted.

The message type ID gives the contents of the rest of the identifier, as associated with the message type (as given Table 5.1). The message type ID acts as a second priority level, since it is the next set of bits to be transmitted. The type IDs have been randomly chosen, but should be relatively easy to re-define if necessary.

The fact that extended identifiers are used also means that a separate protocol should be easy to implement using the standard identifiers (since they are separable). Since CANOpen, DeviceNet, etc. use the standard identifiers, it is conceivable that one of these protocols could run in parallel.

5.3.3 Channels

Channel messages are sent when a channel value is to be updated. The message identifier layout is given in Table 5.2. The 8 bytes of data consist of 32 bits of the value being sent, and a 32 bit time stamp obtained from the source's clock. The presence of the source address (which must be unique) means two messages with the same identifier can't be transmitted on the bus (causing bus failure).

A buffer for each in-channel is currently kept in RAM. This buffer is updated whenever a channel message is received. Functions to read the most recent value, and the time at which it arrived, are available in the library. A function to send a channel is also available (*scandal_send_channel*).

Because many of the channels require scaling[1], a scaling feature was built into the protocol library. A function called *scandal_send_scaled_channel* was added for linear scaling. M and B are stored in EEPROM, and can be updated using the configuration features (to allow for easy calibration).

Channel values must be in units of a thousandth of the respective SI unit[2] (e.g. mA for current, mV for voltage). This means that displays, etc, do not have to perform any secondary scaling, and every channel can be treated in the same manner. Also, fixed point math is used, so (performance intensive) floating point operations are avoided

[1] For example, the ADC output of the DC-DC converter ranges between 0 and 65535, but the value to be output ranges between 0 and 164000.

[2] Two exceptions to this rule are the speed, which is measured in m/h since km/h is the unit normally used to measure vehicle speed, and degrees, which are measured in milli-degrees Celsius for similar reasons.

3: Priority	8: Message Type	8: Destination Addr	10: Parameter Number

Table 5.3: Configuration message identifier

Parameter#	Name	Description
0	Address	Change the address of the node
1	In-channel source	Change an in-channel source
2	Out-channel M	Change an out-channel scaling constant, M
3	Out-channel B	Change an out-channel scaling constant, B

Table 5.4: Configuration parameter descriptions

(which is important on the low performance microcontrollers). Values which have no units (e.g. the state of an indicator) use arbitrary units.

Nodes will not use values from channels with node address of 0 (since this is the default in-channel source value, and the default node-address).

5.3.4 Configuration

Configuration of each node is achieved through the use of explicit, directed configuration messages. The identifier layout is similar to the channel messages. The data layout is different, depending on which parameter is being updated. See Table 5.3. See *scandal_engine.c* for further information regarding the layout of the data section for each message.

Since a destination (rather than a source) address is used in the identifier (in order for the message to be directed to a particular node), it is important that only one node configure a particular parameter at any time. This should not usually be an issue, since configuration is normally performed from a PC, or other master node, of which there is only one. Note that once configuration is complete, it is stored persistently, so no run-time configuration is necessary. This means that if any node fails, the network can continue to operate.

The parameters which are currently configurable are given in Table 5.4.

Configuration message handling is performed in the *handle_scandal* function. It is imagined that user applications will also require configuration. This may need a separate user configuration message, and is not currently implemented. Note that up to 1024 parameters can be addressed, which should be more than enough for all protocol related configuration.

5.3.5 Small Scandal

It became obvious in the writing of the MPPT communications code, that the protocol library outlined in the sections above would prove too large for the limited microcontroller used in the MPPT. Because of this, a cut down version of the protocol, providing only outgoing information, was constructed. Heartbeats and out-channels are implemented. The function names and arguments are identical to that provided in the full Scandal implementation, so upgrading/downgrading to/from the full library is simply a matter of re-linking.

The node address is defined in the *scandal_config.h* file as *DEFAULT_SCANDAL_ADDRESS*.

The small Scandal implementation just fits in the PIC microcontroller, along with the rest of the MPPT code.

5.4 Wireless Communications

Wireless communication is provided via RangeLAN2 wireless Ethernet. A protocol was developed by Luke Macpherson for use over this medium (since a significant packet loss is usually endured with wireless protocols). The *canbridge* program operates as a transparent CAN bridge, implementing the CAN interface previously defined. Thus the support car is essentially another node on the CAN network. The wireless link provides a best-effort connection, but does not exhibit the reliability of CAN due to the inherent unreliability of the physical medium.

5.5 User Interface Software

Two applications were developed for use in configuration, analysis and monitoring of the network. These applications are intended as configuration utilities.

The first resides in a CANRefNode, and interfaces to the user via serial terminal. This program's text interface is capable of sending configuration messages, analysing received packets, displaying channel updates received, and displaying the error status of its CAN controller (useful for determining the health of the bus).

The second is a graphical application which accesses the network via the CAN interface defined in the abstraction layer. Two implementations of this interface have been used — the first communicates using binary serial communications. The second uses the wireless protocol as outlined in Section 5.4.

This second application, Scandalconf, is, as is the terminal application, designed to be an interim solution, or prototype, for a final interface application that is under construction by other students. It is also serving as a prototype for the final program, with different pieces of code cobbled together to perform a variety of interface actions. See figure 5.1.

Scandalconf has been used on three testing runs, and has shown itself to be relatively useful. It is imagined that this application would be used as a fall-back should a more appropriate interface be incomplete at the time of a race. Scandalconf was used to log data for this report. It performs monitoring, configuration, and control functions. It also provides a useful charge integration function (which has been tested successfully).

5.6 Results and Conclusions

The protocol, abstraction layer, and configuration utilities constructed during the project appear to work well, apart from a few significant bugs which remain.

5.6.1 MCP2510

Most of these bugs relate to the MCP2510 CAN controller from Microchip (as used on the CANRefNodes and the MPPTs). As mentioned in Section 4.8.2, the MCP2510 has a number of serious flaws [30]; two of them are particularly debilitating. Errata item 5 states that if three messages are received back-to-back-to-back, the first message can be corrupted because of an overflow condition. This means that any message

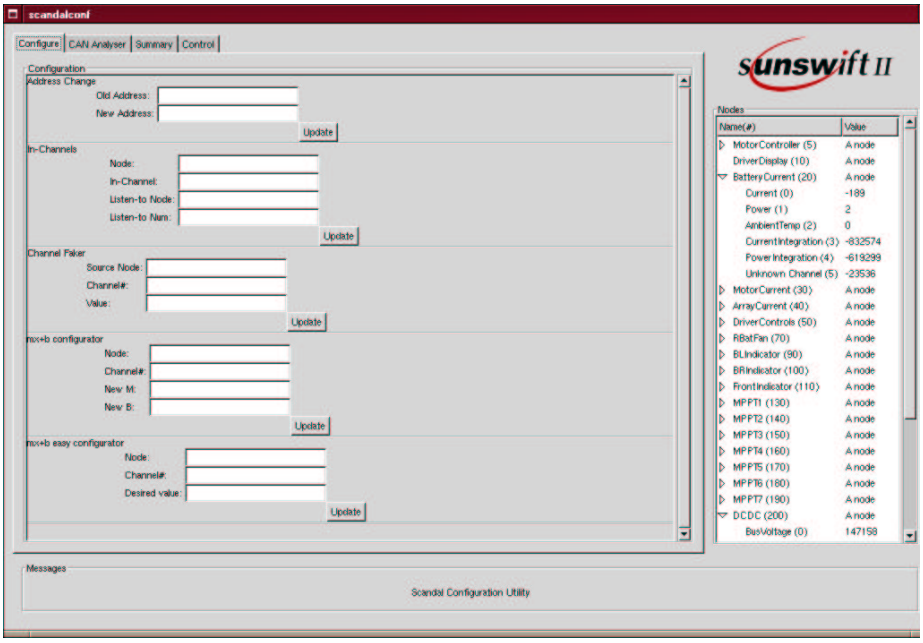


Figure 5.1: Scandalconf configuration utility screenshot

that overflows must be dropped, the result being that CAN can no-longer be considered a reliable protocol. Errata item 6 states that if two messages are received, and one message transmitted, in a short period of time, the transmitted message may be erroneous. Again, meaning that the CAN protocol can no longer be considered reliable.

These two bugs have caused many problems. Garbled messages, which should be highly improbable, given CAN's built-in error checking and re-sending, have appeared, causing control functions such as the indicators to malfunction. These bugs are extremely difficult to trace.

Furthermore, the errata document does not cover all the silicon bugs present in the chip. A further silicon bug has been discovered (and reported to Microchip) during the course of this work. This relates to an error similar to that of errata item 6: erroneous messages appear on the bus. Garbled messages appear when the overflow function is enabled (meaning that a message will be transferred to buffer 1 if buffer 0 is full). This is very limiting because it means that the CAN controller must be serviced far more regularly, as the overflow function must be disabled.

Discussion with a Microchip applications engineer has resulted in their investigation of this issue. They have indicated that a revision to the MCP2510, resolving these issues, will be released in the near future. At that point, all MCP2510 CAN controllers in the car should be replaced with the updated version.

One improvement would be interrupt-based receive functions (where possible), which would reduce latency between the controller's reception of the message and transfer to the microcontroller, clearing the buffers earlier, and reducing the effects of errata item 5.

5.6.2 Conclusions

The software infrastructure has proven extremely easy to use. The defined interfaces provided by the abstraction layer make porting the protocol and other software to new platforms relatively painless. The Tritium code was developed rapidly in this fashion.

As an example of the ease of use, one second year student constructed a one-wire temperature sensor daughterboard and related software. This is a significant project. What's even more significant is that he had designed only one PCB (also for Sunswift) and had never written a C program (his experience was limited to Java). The student has managed to develop a working device which reads and monitors the battery temperatures, based on the CANRefNode, Scandal, and the abstraction libraries.

Chapter 6

Maximum Power Point Trackers

Maximum power point trackers are devices commonly used within photovoltaic systems to optimise the power generated by a solar panel. They are particularly applicable within a solar car where, because of the movement of the car, the environmental conditions rapidly change. Electrical conditions within the car are also highly variable (e.g. as the car encounters hills or accelerates, the motor controller will draw large amounts of energy/current causing the battery voltage to drop).

In June 2001, the UNSW SRT was preparing for the upcoming WSC. New MPPTs were required because the electrical specification for the car had changed (specifically, a new battery pack had been purchased giving a higher upper bus voltage, with which the original MPPTs would not operate). An investigation was conducted into the available products.

A device from Biel University in Switzerland was selected. It was designed and built by the industrial electronics lab at the university. The tracker design was developed for the Solar Motions [1] team (from California) following their negative experiences with the Brusa [6] tracker in the 1999 race.

Unfortunately the MPPT from Biel was not thoroughly tested and debugged at the time of the WSC. While they had been bench tested in Switzerland, extensive outdoor testing had not been possible, much less testing in an on-road environment. While the trackers were operated by some teams in the *American Solar Challenge* (ASC) (a race held three months before the WSC), none of the problems experienced with the trackers were known to the UNSW SRT prior to the purchase of 12 Biel MPPTs. Other teams appear to have had positive experiences with the trackers.

The UNSW team experienced great difficulty with the trackers during WSC and the poor performance of the car during that race can be attributed largely to the malfunction of these devices (as outlined in Section 2.2.2). Many of these issues can be traced to the software running on the MPPT's microcontroller.

This chapter outlines the rework of the software resulting in the trackers' (and car's) successful performance during the 2002 Sunrace event. The addition of various abilities such as IV curve sweeping are investigated, as well as improvements to the tracker's control loop and control software following that event.

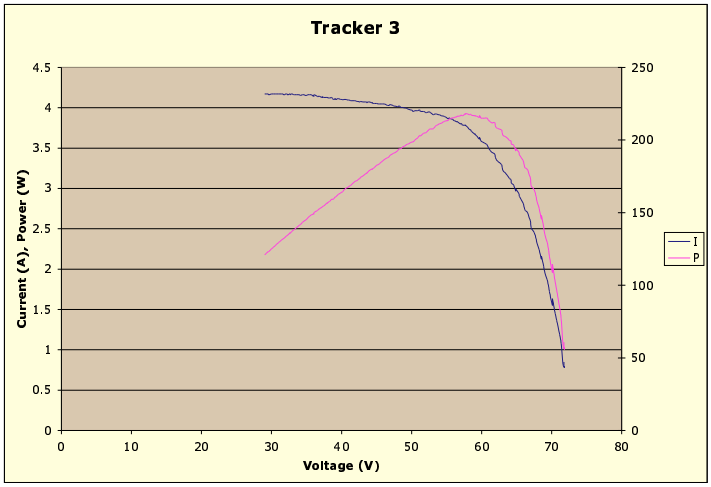


Figure 6.1: Solar panel IV and PV characteristics

6.1 Background

6.1.1 Maximum power point tracking

Photovoltaic solar cells are non-linear devices. The device's I vs V characteristics result in a power-voltage curve with a well defined maximum power (see Figure 6.1). It is obvious from Figure 6.1 that if the cells are operated at a non-optimal voltage, less power than otherwise will be generated[1]. The voltage at which the cell or cells[2] will produce the maximum power (V_{mp}) will change with a variety of environmental conditions: including illumination and temperature.

A tracking mechanism, the MPPT, is employed to optimise the power generated by the solar panel. The tracker draws enough current (I_{mp}) from the input (the solar cell/panel) such that the voltage across it is the maximum power point voltage (V_{mp}). Essentially this style of device performs a DC-to-DC conversion from V_{mp} to the output voltage, which, in the case of a solar car, is the battery-pack voltage.

While the DC-DC converter is not 100% efficient (high efficiency converters can reach 99%), the power gain through tracking the maximum power point offsets this conversion loss. This is particularly true for a solar car's array where the environmental conditions can change rapidly due to the movement of the car, etc.

[1] See Section 6.2.1 for a comparison of optimal and sub-optimal tracking.

[2] Figure 6.1 represents the current and power vs voltage curves of a group of solar cells connected in series (rather than a single cell).

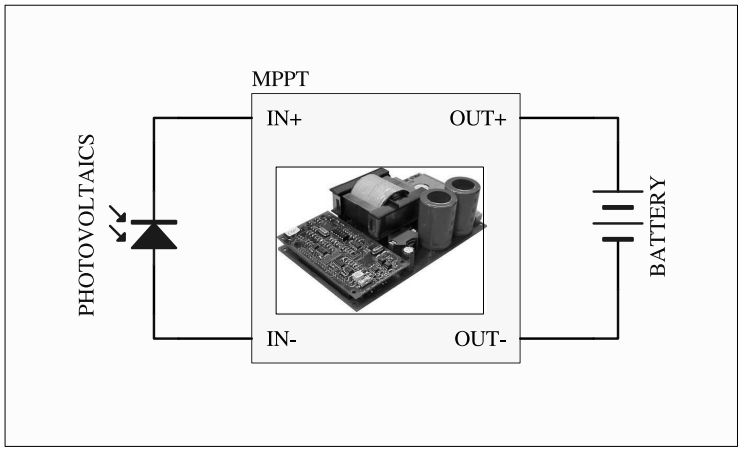


Figure 6.2: MPPT Connection Schematic

6.1.2 Biel MPPT hardware description

The following summary is derived from the "MPPT New Generation Users Manual" [34], the MPPT source code, and schematics.

The new generation tracker, from Biel, is designed around a switching boost converter. A schematic for the basic boost converter topology is given in Figure 6.3. The operation is fairly simple to understand: a PWM signal to the gate of the *field effect transistor* (FET) causes it to switch on and off with a given duty cycle at a particular frequency. When the FET is conducting, the current through the inductor builds up according to $\frac{dI}{dt} = \frac{V}{L}$. When the FET switches off, the current is "forced" into the capacitor through the diode, charging the capacitor according to $\frac{dV}{dt} = \frac{I}{C}$. Thus the voltage is boosted above the input voltage (since $V = L\frac{dI}{dt}$). Varying the FET ''on" time (by varying the PWM duty cycle) varies the current which builds up in the inductor, and thus the voltage at the output.

The Biel tracker uses an extended version of the topology which results in soft-switching. This is also known as zero-voltage, zero-current switching since the FETs have zero voltage across, or zero current through, them when they are switched. This is achieved through the use of an active snubber circuit — requiring an extra FET, inductor, capacitor and diodes. Synchronous rectification is employed to save the voltage drop across the diode: it is replaced with a FET which is switched on when the diode should be in a forward conducting mode. Thus three FETS are required. The control for the gate drivers of these transistors is implemented using discrete logic and passive components. This converter section is extremely efficient compared to other MPPTs used in solar powered cars.

The MPPT is separated into two sections: power and control. The converter, connectors, etc, are located on a main converter board. The control circuitry is situated on a second board connected to the first.

The control circuitry consists of the discrete ICs which implement the control and timing circuitry for the FET gate drivers, and a microcontroller (a PIC 16F877). The microcontroller's ADC is connected to sensors located on the power board which monitor input voltage and current, output voltage, and two temperatures. A switching converter generates the required 15V, and 5V logic circuitry supplies from the tracker's PV

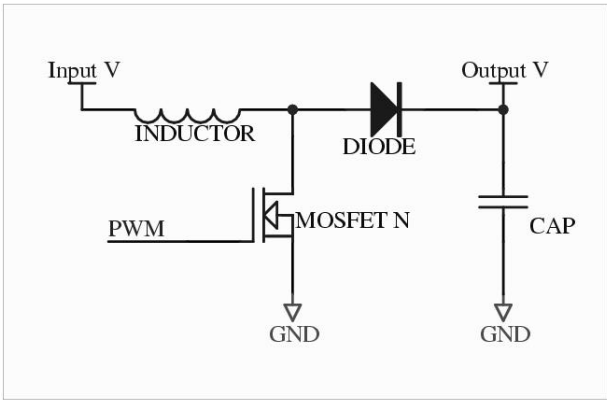


Figure 6.3: Simple boost converter schematic

input.

A CAN interface is provided using an MCP2510 CAN controller interfaced via SPI, optical isolators and a line driver. An RS232 interface is provided using the microcontroller's internal UART and an external RS232 line driver.

The microcontroller has two internal clocks which can be configured to output a PWM signal. These two PWM signals are used to generate the gate drive levels using the logic on the control board.

A number of security mechanisms are built into the trackers, including over voltage, under voltage and over temperature protection circuitry which can shut down the converter module in the case of an error. Fuses protect the input and output.

6.1.3 Tracking algorithms

There are a number of techniques for tracking the maximum power point of an array. These include both analogue and digital algorithms. The methods considered here involve a microprocessor running an appropriate tracking algorithm to control an efficient DC-DC converter. This decision is due to the design of the Biel MPPT (see Section 6.1.2).

A number of novel algorithms have been developed to find the maximum power point, in a device like the Biel tracker. A good comparison of the commonly used algorithms is given by Hohm and Ropp [17]. The target of this paper is to select a maximum power point tracking algorithm for a device very similar to the Biel trackers. The algorithms considered are:

Perturb and observe

The *perturb and observe* (P&O) algorithm is commonly used due to its ease of implementation, and relative tracking efficiency. The idea is that the panel voltage will be continually offset slightly. The resulting change in power output is observed, and the

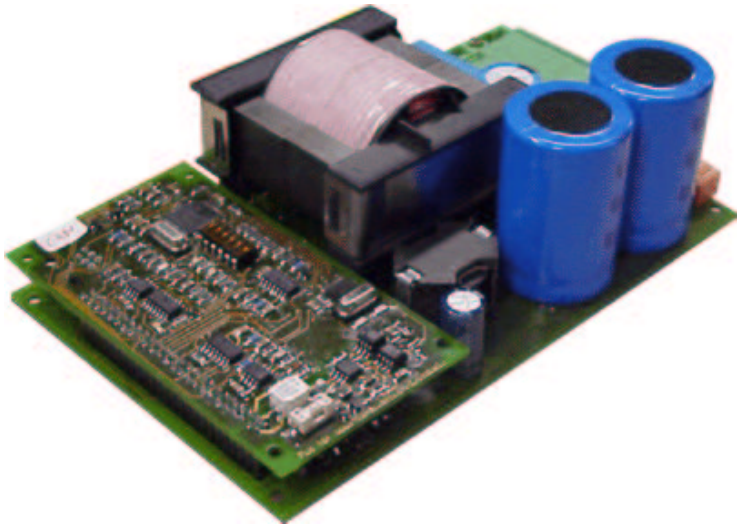


Figure 6.4: Biel Maximum Power Point Tracker

direction of the next perturbation evaluated based on the change in power output. This algorithm is also known as "hill-climbing", for obvious reasons. It is possible for the P&O (and derivative) algorithms to get stuck on a local maximum should the IV curve be non-ideal. The original Biel MPPT software used this algorithm.

Incremental conductance

Hussein et al, in a discussion of the *incremental conductance* (IncCond) algorithm [18], present a technique for overcoming some drawbacks to the P&O algorithm. These drawbacks are given as the steady-state oscillations (as can be observed in Figure 6.5) around the *maximum power point* (MPP), and the tracking inefficiency under rapidly changing atmospheric conditions. It improves the P&O algorithm by using the solar panel's incremental conductance to determine that the panel is at its maximum power point — meaning that the operating point need not be perturbed. Both Hussein and Hohm et al show a significant improvement in the tracking accuracy of incremental conductance over P&O.

Parasitic capacitance

Another algorithm considered by Hohm makes use of the parasitic capacitance of the solar cells. It attempts to improve upon incremental conductance by using the switching ripple to perturb the panel. Unfortunately this method is suited mainly to large solar arrays with specially designed trackers, and is therefore not considered in this work.

Constant voltage

(Also known as "open loop" tracking by Roche et al [40]).

This algorithm makes use of the fact that the ratio of V_{mp}/V_{oc} is approximately constant. Thus, the tracker measures the open-circuit voltage (V_{oc}) by some means, and then sets the input voltage to a proportion of that measured value. Hohm quotes a value

of 78% for this ratio, but the author's experience would give a number between 78% and 82%, with 80% being a good estimate (see Section 6.6). Two MPPTs previously used by the UNSW team have employed this algorithm: those from Australian Energy Research Labs, and those from Northern Territory University.

Conclusions

The incremental conductance technique, when properly implemented, should give the best results of the algorithms outlined above. Hohm and Hussein both give experimental comparisons of the technique to P&O. Hohm also compares to the constant voltage method. P&O and incremental conductance are shown to have better tracking efficiency than the constant voltage technique (by approximately 10%), with incremental conductance slightly better than P&O. It is thought that the advantage of the incremental conductance algorithm over P&O would be more pronounced when used in a solar car environment due to the rapidly changing environmental conditions.

It is possible that techniques not considered above would give slightly better MPP tracking efficiency, but it is unlikely that these could be implemented on the Biel hardware.

6.2 Experiences with the Biel MPPTs during WSC

During the 2001 WSC, UNSW encountered a number of difficulties with the Biel MPPTs.

The first problem was detected on the start line in Darwin, when the trackers failed to start up. This eventually proved to be the over-voltage limiting feature of the units having been set incorrectly (by the manufacturer). The upper battery pack voltage was 164V and the trackers had been set to limit their output at 152V, therefore on the start line with the battery pack at its maximum voltage, the trackers were in their over-voltage mode and did not produce power. This feature should have, but had not, been tested (by the UNSW team) before the start of the race. It was only well into the first day that the battery was sufficiently drained, that the trackers were enabled. This cost the car several kWh.

A second problem was encountered several weeks prior to the race. The Sunswift array is composed of approximately four thousand cells. These cells are then wired in series to form "strings". Each string is approximately 120 cells long, giving a voltage of approximately 60 to 70V when open circuit. In previous races the team had wired one string per tracker, but since the new array had so many more cells, this would have led to over 30 trackers. Since the power through each tracker would have been so small, the poor conversion efficiency (at this low power) would have negated any advantage given by more accurate power point tracking. In order to increase the power through the devices, and reduce their number, several strings were wired to a single tracker. This resulted in a number of combined power curves, which, taking into account the effects of diodes and other panel imperfections, results in multiple maxima — the power points of the strings wired into each tracker did not coincide. This was observed to lead to an overall power loss of several hundred watts when compared with the per-string maximum power.

The Biel tracker used a closed loop algorithm to find V_{mp}. Small adjustments were made to the pulse-width modulation duty cycle and the resulting output power then

observed. Decisions were then made based on the effect of the PWM change. This is the P&O algorithm.

While this was effective for single strings, when multiple maxima are present (due to parallel strings with diodes) it is possible for the tracker to get "stuck" on a local maxima, ignoring the benefits of the global maxima. This effect reduced output power significantly. The device could be led to track a local maxima simply by shading and then un-shading part of the array. This is a common occurrence under racing conditions due to trees lining the sides of the roads, over-passes, passing road-trains, and other sources of intermittent shadow.

During the race this problem was "solved" by manually resetting the trackers after any such event occurred. This led to significant power losses during the morning and afternoon when shadows were more likely to cover the road.

The third, and most serious, problem was experienced by several other teams. Under low light conditions, the tracker would fail in a self-destructive manner. The problem was that battery current flowed back into the tracker and panel. This was protected against following the loss of five of the twelve trackers during the race by placing diodes in series with the output lines, stopping current flowing from the battery into the tracker, but affording a small loss in power (approximately 6W in total).

Other issues include an inability to begin tracking in low light conditions, resulting in a repetitive clicking sound as the tracker repeatedly attempts to start.

6.2.1 Analysis

The software used during the World Solar Challenge was programmed into a tracker, and a number of measurements taken.

Tracking

Figure 6.5 shows the "Perturb and Observe" algorithm in operation. The graph shows the input voltage vs. time for a period of the tracker's operation on a relatively low voltage panel ($\approx 40V_{oc}$). While this exacerbates the effects of the algorithm (since a single bit change in PWM will have a larger effect on the output), it should be noted that several of the panels in Sunswift II will produce significant power at comparable voltages; therefore this is a just real-world test.

It can be seen that there is a large variation in the input voltage on the panel. There was up to 10% variation on the voltage across the solar cells. Thus, despite the high likelihood of the average input voltage being the actual maximum power point, the time the solar panel spends at that optimum voltage is very small, oscillating around it.

It is also possible that these oscillations are set up by via hardware problems. The ADC on board has a significant delay (as was found when attempting to optimise V_{OC} measurements in the new software — see figure 6.8) due to the input filters. This, and other delays in feedback are on the order of the control loop's update period (20ms for 50Hz) resulting in the observed oscillations.

Note that the mean is 30.2V and the *open circuit voltage* (V_{oc}) is 38V. $\frac{30.2}{38} = 0.795$. This indicates that the mean is likely to approximate the maximum power point (as shown in Section 6.6.1).

Further analysis was performed. Equation 6.1 gives the current in an ideal solar cell. Figure 6.6 was generated as a result of the calculation of the percentage error (of the sample from V_{mp}) at each sample point. It can be seen that maximum power is obtained for almost none of the tracker's operation time and periods of time are spent

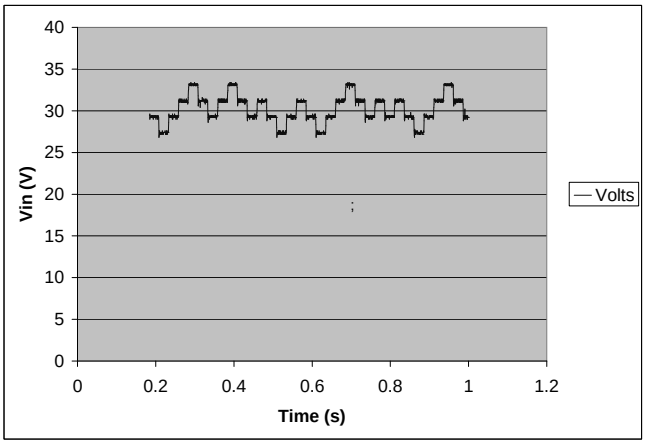


Figure 6.5: Pre-WSC software input voltage vs. time with $V_{oc} \approx 38V$ (1 sun)

$$\begin{aligned} Max &: 33.4 \\ Min &: 26.8 \\ Mean &: 30.2 \\ StdDev &: 1.66 \end{aligned}$$

with a 10% error. The percentage errors were introduced for the ideal solar cell. Figure 6.7 shows the power vs time for this curve as estimated via the percentage error and the ideal solar cell equations.

$$I = Isc - Io * (e^{\frac{qV_{cell}}{kT}} - 1) \tag{6.1}$$

The following figures were calculated (assuming an I_{sc} of 1A) where IC is ideal cell:

$$\begin{aligned} P_{MP,IC} &: 0.496W \\ \overline{P} &: 0.47W \\ \%\Delta &: 4.35\% \\ 1200W\Delta &: 52.1W \end{aligned}$$

Thus, each cell produces approximately 26mW less, a 4.35% change. For a 1200W array, this translates to a 52.1W power loss due to tracking error.

Clicking

The clicking sound heard in the early morning during the World Solar Challenge was observed in a controlled situation. Low input voltages (with limited input currents) were applied using a current controlled power supply (as well as a solar panel under low light conditions). The clicking sound consistently occurred. The proposed explanation is that the WSC tracker sets its PWM duty cycle to a predetermined "start" value, and that start value is too large under low power conditions. The converter pulls the voltage

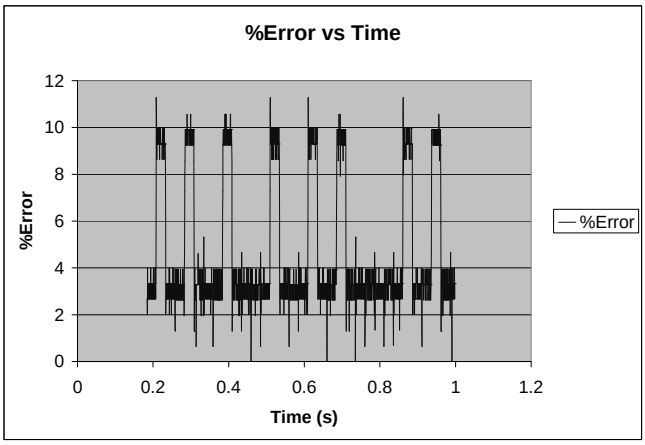


Figure 6.6: Percentage error vs time for the pre-WSC software

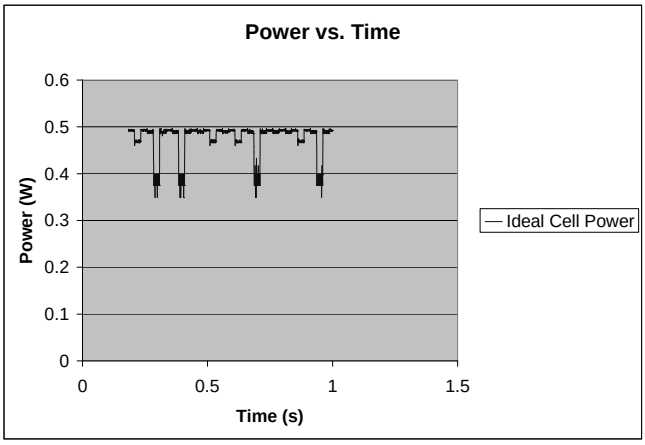


Figure 6.7: Pre-WSC software calculated power per cell vs time

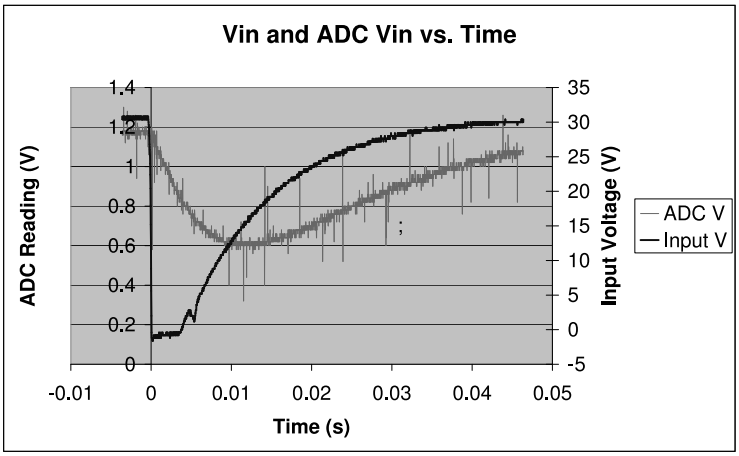


Figure 6.8: Input Voltage and ADC Vin reading vs. Time

on the input capacitor down so fast that the tracker doesn't have time to react (see Figure 6.8). The voltage goes below the shutoff voltage of the power supply for the logic, at which point the microcontroller shuts down, shutting the converter down with it. The input capacitor then recharges and the microcontroller is once again enabled, initialises itself, and the process repeats. The clicking sound itself is assumed to be due to the inductor shaking as the currents through it change rapidly.

Figure 6.9 depicts this process. The delay between the clicks is due to the time taken for the input capacitor to recharge and the tracker to initialise itself, and start conversion.

This process results in a large number of tracker restarts which may have contributed in some way to their failure (possibly through issues caused when the microcontroller browns-out), as well as wasting the available power.

During the World Solar Challenge (2001) the trackers were "jump-started" in the mornings by connecting multiple panels to one tracker. This allowed the tracker to draw enough current such that the input voltage would not dip below the critical level. Once the tracker had started, the extra panels could be removed (since the PWM setting would then change to adjust to the new current).

While modifying the pre-WSC code prior to Sunrace (as outlined in section 6.3), an attempt was made to change the start value to a very small value (assuming that the converter would then find the maximum power point). Unfortunately, this was unsuccessful because the algorithm requires that the power changes as the PWM value changes. If there is no power being produced at all (ie the PWM duty cycle is so high that the panel is virtually open circuit), the algorithm will simply stagnate, still producing no power.

6.3 Sunrace 2002 Improvements

Following the experiences during the WSC, it was decided the software needed to be significantly revised to attempt to solve the issues outlined above. The appropriate development tools were obtained on loan from the Biel labs.

Initial investigations were made regarding the software. The original software was

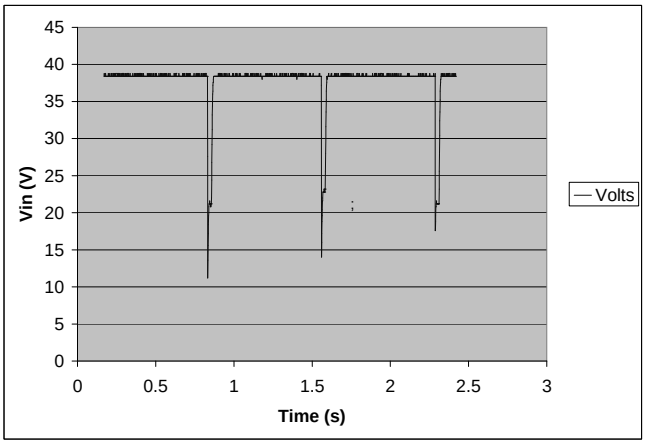


Figure 6.9: Pre-WSC input voltage vs. time in morning light conditions

written in assembly language, which, on the PIC processor (used in the tracker), is extremely difficult to develop and maintain. In order to ensure the trackers would perform for Sunrace (a solar car event, racing from Adelaide to Sydney, held in February, 2002), improvements were made to this original software. Several weeks were spent understanding the hardware interfaces, and function, in the original code (which was in a relatively unreadable form). Serial communications features were added to aid debugging and testing.

Once a usable level of functionality had been achieved (and several problems solved; to be outlined), the software was deemed capable of operating the trackers in Sunrace. At this stage a complete rewrite of the software in C was commenced (since a relatively reliable fall-back was then already available). The compiler used (CCS PCM) allowed rapid development of the new software. Equivalent functionality was obtained after a short period of time.

This rewrite solved many of the issues associated with the trackers. The short-circuiting problem was possibly caused by an error in the initialisation code. The trackers use a synchronous rectification scheme with soft switching. This requires three FETs. If the synchronous rectification FET and either of the other two FETs are on simultaneously, a direct short across the battery is seen. The current developed causes the FETs to fail, also destroying large portions of the drive circuitry. The initialisation code allowed the soft switching FET and synchronous rectification FET to be on at startup. This was normally not a problem since in most circumstances the FETs would be reset almost instantly in another section of the code. The delay to this code was small enough that no current could build up. However, under low light conditions, the system may delay following the initialisation, the current will increase and significant damage will be caused to the tracker. The solution was to set all the FETs off in the initialisation code, only allowing them to switch following rigorous checks (involving the use of a well defined state machine). This theory is slowly gaining support via testing. The trackers have not self-destructed in the same manner since the new code was installed. This is possibly due to the more reliable power supply provided under by new code

(since the tracker will not convert below a minimum voltage). Another possibility is that the brown-out detection circuit was not enabled in the microcontroller. When a low-voltage condition led to the "clicking" problem, the microcontroller would brown-out and the result may have been the destruction of the tracker. This error has not been observed since WSC.

When re-writing in C, many features were available in the compiler; for example, 32-bit operations. This allows for far more readable, functional, code, reducing the risk of mistakes such as the incorrectly set maximum output voltage (experienced during WSC).

The tracking algorithm problem was solved temporarily (for Sunrace) by using to an open loop type algorithm [40]. The tracker is periodically shut down for a short amount of time (causing zero current, and therefore V_{oc}), and the voltage across the panel measured. This voltage is then multiplied by 0.8 (or another constant suited to the particular panel) and the result used as the target voltage for a PWM control loop. This avoids the problem of local maxima (rather than global maxima) tracking. The algorithm results in an efficiency drop since the V_{mp} is not accurately tracked, but a V_{mp} within 1% will result in only a minor power output degradation. The problem should, however, be investigated in the future.

A number of other improvements were made with the rewritten software. A task scheduler was implemented. Serial communication allows a convenient monitoring of the tracker's telemetry data. Further feedback is provided via the LED status indicators. A state machine governs the operating mode.

6.3.1 Optimisations

Open loop measurements

The open loop algorithm has a number of disadvantages which accompany the reliability obtained.

The algorithm deems that measurements of V_{oc} need to be taken at a fixed rate. That rate was set to one sample per 20 seconds for Sunrace. In order to take this measurement, the conversion needs to be shut down by setting the PWM to its minimum value.

So, while the measurement is being taken, very little power is produced. It is, therefore, very important that the tracker be disabled for as little time as possible.

A number of experiments were conducted. It was realised that varying light conditions would affect the time required for the input capacitor to charge to the open circuit voltage. A small loop was written to continually check for zero current on the input, forming a delay. It was found, however, that for some unknown reason the loop failed to detect zero current correctly (later determined to be associated with the ADC input filters). The software from Sunrace 2002 used an overly generous fixed time delay. Under most conditions this was sub-optimal, since the input capacitor will charge in a shorter time than the fixed time delay (see Figure 6.11).

Control loop implementation

It was noticed that in the original code, the input voltage would oscillate. This was partly because the original code made constant size step changes to the PWM duty cycle (which controls the conversion rate). By introducing a PWM control loop with

the input voltage delta as the input parameter, and the PWM delta as the output, the oscillations were reduced significantly.

The control loop implemented was intended to be a generic P type control loop[3]. Following the race, and further investigation regarding some anomalies observed at the output, it was discovered that a bug in the code made the P type control loop an I type control loop[4]. This results in a loop which is less responsive, but exhibits no steady-state error.

The loop was tuned experimentally. This proved very effective. Some overshoot was observed and to prevent the tracker from overshooting so far (to such a low voltage) that the logic power supply would fail, the proportionality constant was reduced. This further reduces the tracking speed but provides for better reliability in low light conditions (when small PWM changes can have a large effect on the input voltage).

It is envisioned that should a closed-loop algorithm be implemented, it would form a second (outer) control loop with the target voltage as an output given to the inner control loop.

6.3.2 Sunrace errata

The devices performed very well for the duration of Sunrace. One problem was noticed by team members — at the end of charge (when the output voltage reaches its predefined maximum), the trackers are expected to back the input voltage, towards V_{oc} in order to reduce the current and thus control the output (battery) voltage. Some erratic behaviour was noticed during operation in this mode. (e.g. when the battery-pack voltage was at its maximum).

This is very likely due to the poor implementation of the end-of-charge control loop. The Sunrace code used a simple PWM decrement to back the converter off resulting in very poor control. A parallel control loop could be implemented with the output voltage used to calculate the delta (rather than the input voltage). The tracker could then choose between the two control loops depending on whether the output is over-voltage or not.

6.3.3 Analysis

Tracking

For comparison with the previously used software, the tracking algorithm was observed. As was mentioned in Section 6.1.3, $V_{mp} \approx 0.8V_{oc}$. the algorithm in the Sunrace software makes this assumption, measures the open circuit voltage and runs a control loop with integral (I) feedback to track that input voltage[5]. Thus, the trackers will operate unless the V_{oc} is less than the minimum allowed tracking voltage (at which point the tracker will idle until this condition changes or the logic power supply is shut down). This also solves the clicking problem, since the initial PWM setting is at its minimum (ie no current drawn).

Figure 6.10 depicts the input voltage control after the Sunrace software had been installed (with very similar conditions to those with which Figure 6.5 was taken). The

[3] A proportional control loop is where the output delta is determined via direct proportionality with the input delta.

[4] An I type control loop is one in which the controlled value is proportional to the integral of the input delta.

[5] Note that this allows a maximum and minimum tracking voltage to be set. If the estimated V_{mp} is lower than the minimum allowed for the tracker, the control-loop target voltage will be clipped at that minimum.

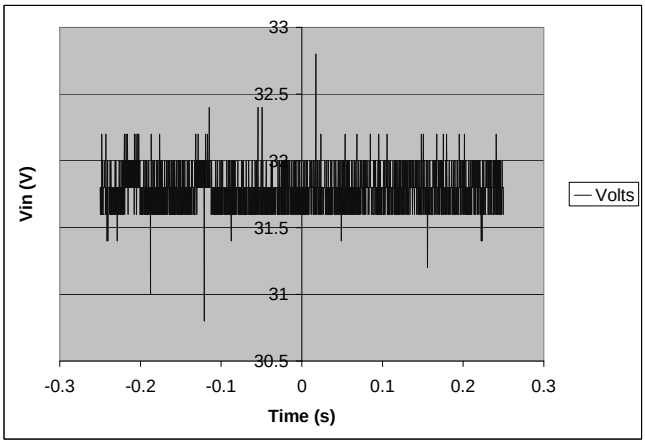


Figure 6.10: Dubbo code input voltage vs Time (1.59A)

$$\begin{aligned} Max &: 32.8 \\ Min &: 30.8 \\ Mean &: 31.8 \\ StdDev &: 0.146 \end{aligned}$$

open-circuit voltage in this case was measured$\approx 40V$, giving $V_{mp} \approx 32V$ using the 80% assumption. Therefore there was an 0.6% error. Also note that the standard deviation of the voltage in the figure is on the same order as the resolution of the ADC (0.104V).

It has been assumed, for the purpose of these analyses, that the conditions remain constant enough that rapid changes in V_{mp} do not affect the tracking efficiency. This is certainly not true for a solar car's photovoltaic array (where movement provides large insolation variation).

6.4 November 2002 Improvements

Since Sunrace, the trackers have developed in parallel with the other hardware and software infrastructure elements of the project. A number of specific improvements have been attempted.

6.4.1 Multiple tracking algorithms

Code to switch between multiple MPPT algorithms was developed. Currently the algorithms implemented are the original open-loop method, and an IV sweep function which is implemented as a second tracking algorithm.

The tracking algorithm can be changed via a serial interface.

It is envisaged by the author that several tracking algorithms be implemented and compared. Other parts of the project overran their allotted time during this project,

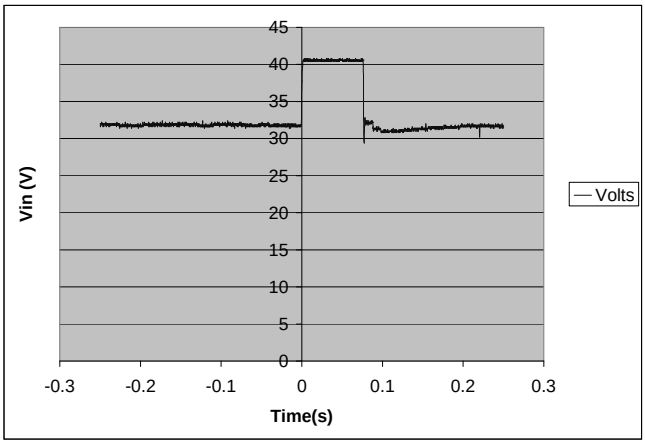


Figure 6.11: Dubbo code finding V_{oc}

meaning that this particular part was not completed.

Interesting tracking algorithms include a re-implementation of the P&O algorithm as a loop controlling the tracker's target voltage rather than PWM, and incremental conductance, which is similar.

6.4.2 IV curve sweeps

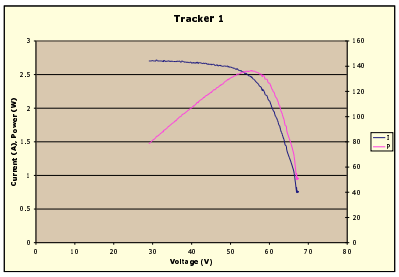
In order to assess, debug, and optimise a solar array, an IV curve sweeping tool is invaluable. An IV curve sweeper measures current vs. voltage between 0V and V_{oc} in a solar panel. Faults with solar cells can be observed graphically. The maximum power point can be determined via analysis of the IV information.

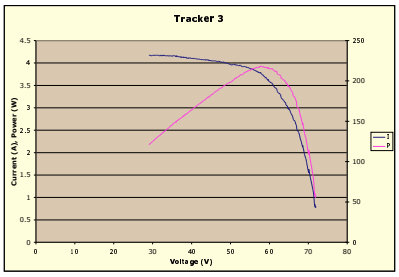
IV curve sweeping functionality was implemented in the MPPT. This is implemented as a second tracking algorithm, which changes the voltage from V_{oc} to the minimum tracking voltage, and outputs IV information to the tracker's serial port as this sweep occurs. Thus, all that should be required to obtain the IV curve seen by a tracker is to connect a serial cable and request a sweep.

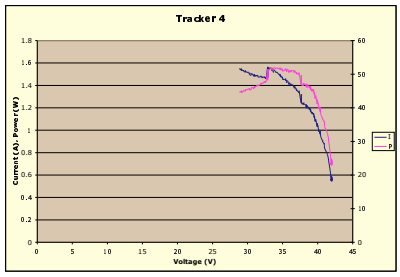
This function has been used by the students working on the car's solar array to obtain a set of IV sweeps for Sunswift's array, and determine a maximum power output. It has also been used to determine a maximum power output for the Sunswift I solar array, which was required for its valuation and sale. The IV curves for Sunswift II's seven panels are given in Figure 6.12.

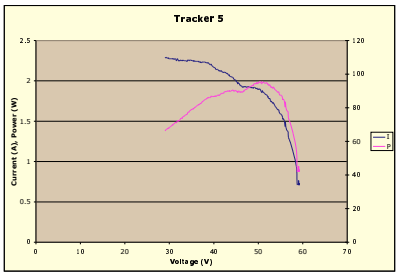
6.4.3 CAN communications

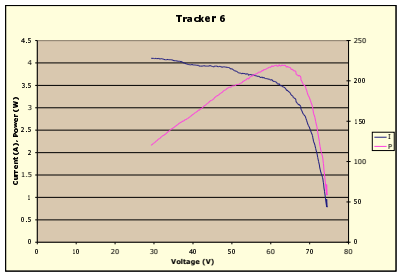
One significant addition to the tracker's code base was the introduction of the Scandal protocol (as outlined in Section 5.3). Since this protocol was designed to be portable, the integration of this software should not have been a significant challenge. However, a severe lack of flash memory in the tracker's microcontroller proved to be a severe obstacle.











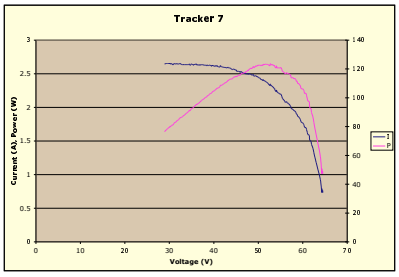


Figure 6.12: Power and Current vs. Voltage per Panel, taken 13th October, 2002

The full Scandal protocol, along with the existing tracker code, well exceeded the 8kB flash memory. The serial protocol, IV curve sweeping code, and other superfluous functions were removed, but the code still remained too big for the microcontroller.

It was at this point that a limited version of the Scandal protocol was implemented (as outlined in Section 5.3.5). This version of the protocol had only telemetry sending abilities, with no message-receive functions used. The abstraction layer functions were implemented, and the same MCP2510 CAN controller code as used for the CANRefNode compiled.

With this extremely limited software, the tracker code will fit in the microcontroller's program memory. Unfortunately, this limitation carries over to the functionality of the code: only one tracking algorithm can be used, IV sweeps can not be taken over the serial port, much less requested over CAN, etc. This is a frustrating situation, since the code exists and is operational.

Two code versions have been built and used. One with CAN support, and another with IV sweeps and a serial interface. It is hoped that further optimisation of the code might reduce the footprint allowing for more features to be included in the CAN version (which would be used during races).

The CAN communications code sends status and telemetry data via heartbeat and channel messages. Three telemetry channels are available: panel voltage, panel current and output voltage. Ambient temperature, heatsink temperature, and open circuit voltage are available, should more program space become available on the microcontroller for the inclusion of the code required to send these values. Heartbeat messages inform the rest of the system of the tracker's operation, along with minimal status information.

Figure 6.13 shows data logged during a testing run on October 27th from Pheasants Nest to Sutton Forest on the M5 motorway (which leads to Canberra from Sydney) commencing at approximately 3:00PM. This data was logged via the wireless Ethernet connection present in the car (see Chapter 8).

One significant issue was experienced while testing this software in the car — the MPPT hardware is extremely sensitive to variations on the 5V power rail. When the voltage is too low (e.g. 4.5V), the tracker will fail in a mode which will disable the CAN network. This brown-out is avoided in the current implementation via the addition of a 5V regulator at each end of the bus, which holds the voltage at 5V using the 12V rail as a source.

6.4.4 ADC interrupt driven control loop

One significant limitation of the Sunrace software was the very slow response of the control loop. The response was delayed such that if the voltage did go below the minimum tracking voltage for some reason, the microcontroller would not respond in time (see Figure 6.8).

This was affected by two significant factors. Firstly, the *low pass filters* (LPFs) on the ADC inputs, designed to remove noise created by the FETs' switching, introduced a significant delay. It is thought these filters were mis-designed, and that the cutoff frequency for these LPFs was far lower than had been desired (approximately 5Hz as opposed to 100Hz). Furthermore, there is little need for such harsh cutoffs, since the ADC within the PIC 16F877 microcontroller is capable of sampling at approximately 15kHz. It is thought that oversampling the channels on the ADC, and executing some form of digital filter within the microcontroller will prove to be more effective than the current passive RC LPFs, and will provide for a much faster response.

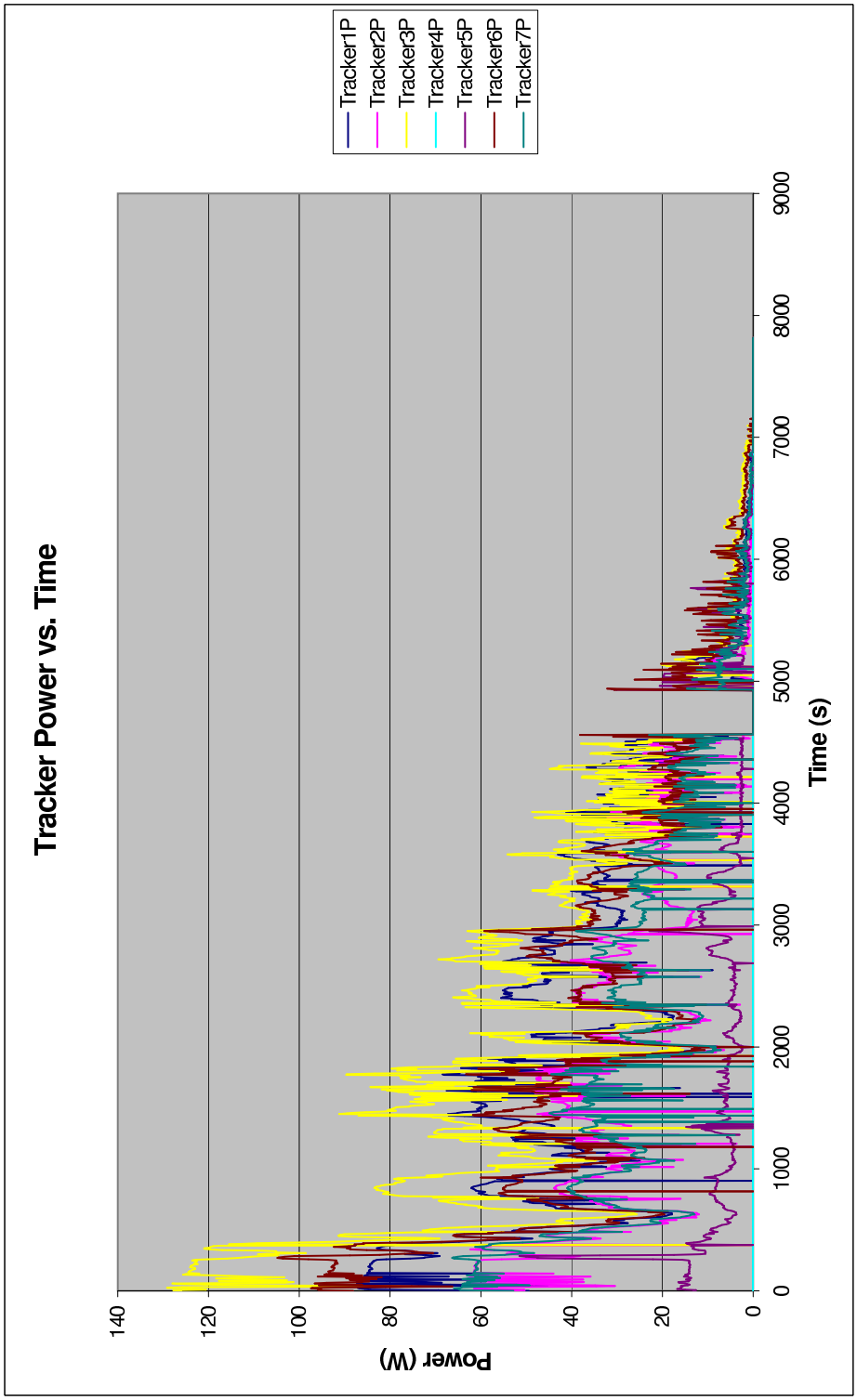


Figure 6.13: Log of MPPT data taken during October 27th testing run

The delay introduced by the LPFs is illustrated in Figure 6.8 and gives rise to the second reason for the slow response. It is impossible to run the control loop at a higher speed than approximately 20Hz, since the loop rapidly becomes unstable (assumably due to the delay introduced by the LPFs). Furthermore, the Sunrace code's architecture prohibits a high speed voltage control loop, since all control functions are performed in the main loop alongside communications functions. The situation is far from ideal, since the control loop code is not guaranteed to run at the specified time interval (it is not deterministic).

To solve the latter issue, an attempt was made to run the voltage control loop in the ADC interrupt routine. The ADC conversions should take approximately constant time, and therefore provide a regular source of interrupts in which to run the control loop code. The ADC filters were removed entirely in order to allow oversampling and filtering within the microcontroller. 32 samples of each parameter were taken between control loop iterations, giving a control loop update rate of just less than 500Hz.

Unfortunately, time ran out for tuning the control loop, but it is thought that this version of the software will provide much faster tracking. It is also thought that the software would be capable of updating the PWM duty cycle at a fast enough rate that the tracker could avoid failure conditions such as the clicking problem.

A high speed control loop would also allow for rapid variations in solar insolation, as well as allow further control loops controlling the desired tracking voltage (for example P&O, or incremental conductance).

Further issues may be experienced with the number of interrupts exceeding the performance capability of the microcontroller. This should be investigated.

6.5 Testing

In order to prove the reliability of the new software operating on the trackers, the device has been tested using a variety of means:

6.5.1 Bench testing

During the development of the revised software, bench testing and tuning was performed. A current controlled power supply was used to supply the input of the tracker, and a lead-acid battery pack was used as the load. A resistive load (in the form of a 60W light globe) was also used, but the power supply current required adjustment such that the output voltage did not exceed the tracker's output limit.

A wide range of input and output voltages were tested, along with a wide range of input currents. The control loop appears to be stable under all of these conditions.

Further testing in this manner was performed when the team's race battery pack needed to be charged. Unfortunately, the power supply has a maximum output voltage of 150V, and the pack's top-end voltage is 164V. A tracker, with a constant voltage tracking algorithm, was used to boost the voltage from 100V (the tracker's maximum input voltage is 105V) to the pack voltage in order to charge it. Thus several hours with 6A input current were administered. While the heatsink and inductor did warm, the tracker still functioned well — its conversion efficiency measured at approximately 96% in these conditions. This charging scenario has been repeated several times.

During the course of bench-testing the trackers were abused in a number of ways as a result of accident. Including shorting the input terminals and shorting the output terminals. The tracker involved has not been damaged by this abuse.

6.5.2 Initial PV testing

Two commercial solar panels were purchased during the 2001 WSC in order to test the trackers. These panels were used to perform initial testing of the tracker with a photovoltaic system. It was using these panels that the IV curve sweeping software was developed, and the CAN communications code was tested. The limitation of any solar system is, of course, that it is only usable depending on the weather and time of day. This meant that code was usually developed using the power supply, and then tested using this PV system. Note that this PV panel is made up of 72 cells in series, with no parallel strings. Therefore it is not a good simulator for the current Sunswift II array which has more cells in series, and several strings in parallel for every tracker. For this reason, in-car testing was favoured.

6.5.3 In-car testing

The tracker software has been tested on a number of occasions in the solar car itself. This is by far the most rigorous testing environment. Unfortunately, it is also the most difficult to diagnose and monitor (or, more correctly, was, prior to the use of the CAN network), since physical access to the device is inconvenient while the car is in motion.

The software has been installed in the car since the preparation for Sunrace 2002, being reprogrammed as the software was updated. The car was driven during Sunrace 2002, with positive reports from the team. Six on-road testing days were held more recently in order to test the CAN network, and the trackers were operational on these days.

The trackers and tracker software have performed well through all of these testing periods, and have a total of approximately two weeks on-road experience, plus much more on the bench. Lastly, IV sweeps of both Sunswift I and Sunswift II have been taken, and the tracker was used for this purpose. No obvious problems have been reported.

Further in-car testing will prove more useful since the CAN network is now available.

6.5.4 Conclusions

Through all of this bench testing (as well as the initial PV testing as outlined below), no protection diode was added to stop the problem with reversed current outlined in Section 6.2. It is the opinion of the author that this problem has been solved through the software re-write, although since the exact cause of the problem has not been identified, the error may remain.

The problem observed in the WSC, where the tracker would get stuck on a local MPP, does not occur with the constant voltage tracking algorithm, although the overall efficiency of the tracker is decreased.

The trackers have not malfunctioned in any of the conditions to which they were subjected. Given that the MPPT has now been shown to be relatively reliable, the focus can be shifted to the efficiency of the device (while not compromising its reliability).

Tracker#	$V_{mp}/V_{oc}\%$
1	82.5
2	78.6
3	80.72
4	82.2
5	85.2
6	83.5
7	81.6

Table 6.1: V_{mp} as a percentage of V_{oc} for each tracker: gathered 13th October, 2002

6.6 Results and Conclusions

6.6.1 Open-loop algorithm evaluation

Table 6.1 represents the V_{mp} as a percentage of V_{oc} for each tracker's panel in the data gathered on 13th October (which was used to generate the graphs in Figure 6.12). This shows the MPP to be relatively close to 80%, and thus the assumption of the constant-voltage tracking algorithm to be relatively correct. Further calculations based on this data gives a worst case tracking error of 1.16% for the Sunswift II array, not accounting for temperature and solar variation, as well as the energy lost due to the V_{oc} measurements.

These results are reasonable, and tolerable given the extra reliability afforded via the new software. The error can be reduced by setting a specific V_{oc} percentage to use for each tracker. Alternately, a full IV sweep could be performed by the tracker in order to determine the V_{mp}.

It is possible to improve the tracking efficiency substantially through the implementation of a P&O, incremental conductance or other continuous conversion algorithm.

6.6.2 Control board deficiencies

The logic and control section of the trackers has been found to be limiting the performance of the tracker in that:

- filters on the ADC input lines cause a significant delay to the signals being measured;
- the ADC is not of sufficient accuracy or precision. The 10 bit ADC gives rise to 0.104V input voltage resolution, 0.15V output voltage resolution and 0.01A input current resolution (for the UNSW configuration of maximum values of 105, 164 and 8 respectively). The 0.104V input-voltage resolution is particularly limiting since this is the feedback to the control loop. Note that the standard deviation of the input voltage while tracking is similar to the input voltage sense resolution;
- the PWM resolution is insufficient. Under some conditions, a 1-bit (of 1024 bits) change to the PWM setting resulted in a significant change in the input tracking voltage. This is particularly true in low light, or low voltage conditions (such as when a small panel is connected to the tracker);

- the discrete logic associated with the control module is currently hard-wired. It might be beneficial to build this into a programmable logic device (improving flexibility, upgradeability and footprint size);
- the PIC microcontrollers are designed to be programmed using assembly. While C compilers exist for this architecture, there are microcontrollers available which are more suited to development using C code. Atmel's AVR range is a good example;
- the PIC microcontrollers use more power, for a lower performance, when compared with many other available devices. One good example is the MSP430 range from Texas Instruments, which uses approximately 1/16th of the power for a higher performance device (a 16-bit microcontroller);
- the microcontroller runs on a 5V supply. It would not be difficult to run a 3.3V supply for the microcontroller and associated logic, cutting power consumption of these devices by a significant amount;
- the microcontroller is programmed very slowly when compared to competing devices (approximately 2 minutes vs. an Atmel ATMega323's 10s). This limits the speed of software development and debugging;
- the CAN controller used has a number of debilitating silicon bugs and should be replaced (possibly with the Philips SJA1000, or better, a microcontroller with a built-in CAN peripheral);
- no real-time clock is available for accurate time-stamping of telemetry data;
- insufficient debugging LEDs are present on the board;
- the connectors for the CAN bus do not match Sunswift's CAN connector standard.

6.7 Future Work

6.7.1 Logic re-design

Possibly the most pressing issue with the trackers is the lack of program memory available on the microcontroller used. This has meant that the tracker's operating software has been fragmented into several smaller programs which each fit into the program space. Little further development can be performed without an increase in this memory.

If the logic board were re-designed, the issues associated with this module, outlined in Section 6.6, could be addressed and resolved. The power consumption could be reduced via careful design, and the device made physically smaller.

Should this be considered, the manufacturer, Biel, should be consulted such that work is not duplicated.

6.7.2 Software improvements

Should the control section be improved, several further features with regard to the software can be developed:

- A number of MPPT algorithms can be investigated and assessed. The trackers (assuming a modified control section with a larger microcontroller) form an excellent base for experiments in this field. Multiple MPP algorithms can be implemented (possibly using the framework already in place). Information and directives obtained via the CAN network could be used to select the algorithm.
- Full integration of the various pieces of software written could be achieved. This would allow transmission of the IV curve sweeps (obtained using the code already written). Further features using the network, such as global power output limiting (for end-of-charge back-off), could be implemented.
- Microcontroller and logic power consumption could be investigated. This becomes particularly relevant as the number of trackers increases, and the efficiency of the power section leads to the logic power becoming a larger part of the tracker's overall energy budget. Measures can be taken in software to reduce this cost through the use of sleep mode, etc.
- Configuration of the tracker could be performed over CAN or the serial connection, easing the tuning of the control loop, changing CAN address, etc.

6.7.3 MPPT re-design

The most beneficial action, in terms of array power output, with regard to the MPPTs, is the use of per-string trackers. The current MPPT's and array specifications are poorly matched, since the devices are optimised for larger solar panels than those which Sunswift's array is divided into. Connecting panels in parallel was necessary to allow currents and powers at which the trackers would operate efficiently. Trackers capable of operating on a single one of the thirty-two strings would be far more effective. This would require a physically and electrically small tracker with low current draw.

Such a device is currently under development by Biel, using a resonant sepic converter topology. A deal has been negotiated whereby the UNSW team would be able to obtain 40 of these trackers at little to no cost.

The power section of the new tracker is being designed with an identical logic-board interface. This means that any development of logic, or software, for the current MPPTs, should operate with the new devices.

A further consideration is that to reduce logic losses and tracker cost, it may be beneficial to build two or more converter modules into a single device (using one logic module). This would reduce size, weight, cost and overall power consumption, of the new MPPT system. This would require further negotiations with Biel, or the construction of such a device locally.

Chapter 7

Motor Controllers

The motor controller is a device used in Sunswift II in conjunction with both the CSIRO wheel motor and the T-Flux motor manufactured by Paul Lillington. Both of these are three phase brushless DC motors. The motor controller performs the function of a commutator. See the Speed of Light books [11, 40] for a good summary of commonly used approaches to a solar car's drive system.

Motor controllers are notoriously unreliable. Even Honda, in 1996, experienced a motor controller failure. Sunswift has a number of motor controllers as spares because of this.

7.1 Interfacing

The motor controller presents a difficult problem in interfacing. This is because the UNSW team owns a large number of different kinds of motor controllers:

- Lillingon controller — manufactured by Paul Lillington, for use with his T-Flux motor. Sunswift owns two, one of which was damaged during testing in 2001;
- UQM controller — manufactured by Unique Mobility. Controller specified for use with the CSIRO wheel motor. Has been used by Sunswift in every event in

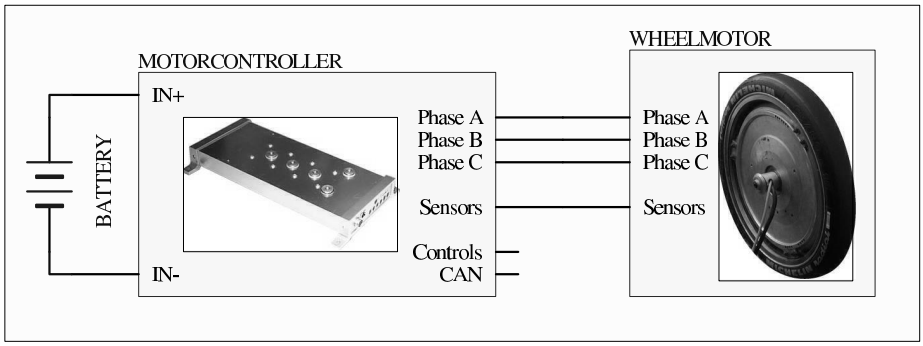


Figure 7.1: Motor controller connection schematic

which it has competed, and has therefore driven approximately 20,000km (with only one failure during this period);

- NTU controller — manufactured by Northern Territory University. Originally designed for use with NTU's own motor (on which the NGM [10] design is based), and adapted by NTU for use with Sunswift's CSIRO motor;
- Sunshark controller — manufactured by the University of Queensland Sunshark team for their 1999 WSC campaign. The team has access to one of these;
- Tritium controller — manufactured by ex members of the Sunshark team comercialising their PhD research. The Tritium controller is a recent design. The team owns one of these controllers and has been promised access to a second controller during races.

Each of these controllers has a slightly different interface. The Lillington, UQM, and NTU controllers expect a control voltage as a throttle control signal. This signal is interpreted in different ways by each controller. The standard approach is to use a potentiometer configured as a resistor-divider in order to generate the voltage and control the motor controller and therefore car's speed. However even within these three, the voltage is interpreted differently. The UQM translates a negative voltage into a command to go into reverse. The NTU controller translates it into a regenerative braking mode. The Lillington does not tolerate a negative voltage on the input.

The Sunshark and Tritium controllers both use a digital interface, although, again, it is very different. The Sunshark controller uses a PWM signal, and the Tritium uses a point-to-point RS485 link with a set of intelligent driver controls.

It is not practical to change the driver controls mid-race, if the motor controller fails and requires replacement. The motor controller itself can be swapped in approximately 30 seconds, whereas the driver controls would require an extended period beside the road.

The solution used during the 1999 and 2001 WSCs was to develop a standard interface to which each controller could be adapted. A 15 pin connector was used to supply analogue control voltages to be interpreted by the controllers. Two adapter boards, for the Sunshark and Tritium controllers, were developed in order to convert this interface to the digital interface expected by the controller itself. These two boards were tested prior to the 2001 WSC, but failed during the race.

Modifications were made to the NTU, UQM and Lillington controllers to allow them to interface to the same driver controls.

Changes became a matter of disconnecting cables, replacing the device, and reconnecting the cables to the new controller - taking approximately 30 seconds.

A number of deficiencies were observed during the 2001 WSC and 2002 Sunrace events.

First, none of the telemetry features of any of the motor controllers were used. The Tritium transmits 15 telemetry channels over the RS485 link, the UQM has a number of analogue voltages whose magnitude correspond to various parameters, the Sunshark controller has similar analogue voltages available.

Second, the analogue control electronics, in attempting to be generic, was difficult to design. Compromises were made. A good example was caused by the UQM's power supply, given to the driver controls, which had a high line impedance ($2.2k\Omega$). The electronics within the driver controls drew enough current that the voltage would drop from 15V to 11V across the supply line. This, in turn, meant that the UQM's highest

control voltage was limited, leading to the situation where the maximum throttle setting was limited to 60% of what it should have been.

Several bugs were observed with regard to regenerative braking. The controls would cause the car to brake very slowly or very severely, seemingly randomly alternating between the two.

Third, there was no facility to connect to the CAN network being developed as part of this project.

A fourth issue exists in the nature of the controls behaviour. Because a potentiometer was used, it was possible for the motor controller to be switched on with the throttle full on. A digital encoder style interface not only eliminates this problem, but also improves the feel of the throttle control. This is important since the throttle is the main control for the car's power usage, and therefore its ease of use is critical.

An interfacing solution was required which provided a common interface for all the motor controllers. To interface the Tritium and Sunshark controllers to the old analogue controls would require further development of the analogue-to-digital adapter boards. This would encumber the system with many of the problems outlined above.

A driver interface board is sold by Tritium as part of their motor controller package.

Since the driver interface needed to be rebuilt to improve its reliability, the decision was made to go for an all-digital system. Adapter boards for the controllers can be manufactured which convert the digital (instead of the analogue) interface to the controller's specific requirement. Since the team's primary motor controller is to be the Tritium, and this controller provides all the features (including a CAN interface) that are required, the decision was made that its digital interface should be standardised upon. In this way, a driver controls board is already available, along with a usable controller which can immediately be installed in the car. The team gets the benefit of the testing performed by Tritium regarding its driver controls, and associated protocol. The source code used in the Tritium controller is available and therefore only a simple port to whatever adapter board is used should be required.

Since the NTU and Lillington controllers will not operate with the car's current battery pack, adapter boards were developed for the UQM and Sunshark controllers only. The adapter boards provide a very similar interface to that on the Tritium controller, with the same connectors, and providing the same features. RS485 and CAN interfaces are provided to connect to the driver controls and network respectively.

These two adapter boards are very similar, with only the controller interface circuit differing (an digital-to-analogue converter is used to control the UQM's analogue interface, whereas a PWM signal is provided to the Sunshark controller). These were co-developed with a student from the Sunswift team (Kian Chin) but have not yet been tested because the operating code has not been written. This is likely to be completed over the coming summer break. No adapter board is, of course, required to interface the driver controls with the Tritium controller.

7.2 Tritium Motor Controller

In order to interface the Tritium controller to the network within Sunswift, both interfacing hardware, and appropriate software were required.

An attempt was made to, as far as possible, adapt the Tritium controller to conform to the guidelines given in Chapter 4. The controller provides an un-isolated CAN interface with a single Microfit connector (ie not the same as that used in the rest of

the network). Further hardware issues relate to the available LEDs, and the lack of an accurate real-time clock.

Two options were considered (and eventually implemented). The line driver was removed and the logic level signals brought to the connector. A small board was then designed with an optical isolator, Triad connector, transient voltage suppression, and appropriate regulators for the power supply[1].

A further board was designed to replace the internal expansion card, negating the need for an external adapter. This board was never built because the small adapter board worked well. Tritium have been made aware of the issues, and indicated that they will route the CAN logic level signals to the expansion card in future revisions of the motor controller. This would mean that the adapter circuitry could reside within the controller's casing, and the connectors mounted on the front panel — a neater solution.

Writing software for power electronics such as the motor controller or MPPTs is usually difficult due to the risk of causing damage to the device itself. Changes in the interrupt timing, accidental modifications to control parameters, garbled memory, or a garbled program, could all have a devastating effect. A current limited power supply was used whenever testing new software in order to limit the effects of any errors. Three controllers were damaged during the period of the thesis due to bugs in software, hardware problems, and operator error. These controllers were kindly repaired by Tritium.

In order to minimise the risk associated with writing this code, it was developed in close consultation with Tritium by travelling to Queensland. This allowed them to give advice regarding implementation techniques, and share their experience with the TMS320L2407 DSP (on which the controller is based).

Code was written to implement the CAN interface as defined by the abstraction libraries. A timer was emulated using the PWM interrupt service routine, and other required functions, such as those for persistent storage, were temporarily stubbed in. Scandal function calls was then used in the motor controller code, and the libraries compiled in.

This process took approximately two days, with almost all of the work relating to writing correct CAN controller code. All telemetry channels available over the driver controls' RS485 link were made available as Scandal channels. This is approximately 17 different pieces of information. The speed, PWM, and motor current are updated at 0.1s intervals. Other channels are updated once per second.

The controller was tested on a number of occasions, failing several times due to causes unrelated to the new code.

The facility to run a speed control loop was developed (by Tritium). Unfortunately, there was no way for their driver controls to control the set-speed. The software was written such that the driver controls board could enable and disable the cruise control function. The set speed and current limit could then be set via the CAN network (using Scandal in-channels). With very little bench testing, this was program was not trusted. It is intended that the code should be further tested and verified, and then be road-tested. The ability to set the speed from a support car will greatly improve the control the strategist has over the car. It should also eliminate inefficient power and speed behaviours as demonstrated in Figure 2.3.

[1] The controller provides a 15V power rail on its CAN connector, so a simple LDO regulator was required in place of an isolated DC-DC converter.

Chapter 8

Sunswift II Implementation

In order to form a working electrical system for use in the solar car, a large amount of work was required beyond the development of the basic infrastructure. A number of devices were designed, based on the CANRefNode. The operating software for these devices was written. The boards were sent to be manufactured, and then populated by members of the solar car team, almost all of whom were in their first year of university. Some design work was carried out by two other students.

The solar car (as mentioned in Section 2.4.1) comprises two halves: top and bottom shell. Since these two components separate, it is necessary for the CAN bus to encircle both in order to service all devices.

The most important parameters to be monitored are battery voltage, array, battery and motor current, and speed. While battery voltage is provided by the controller and trackers, a high accuracy value was required which these devices could not provide. While it is possible to deduce the battery current using the sum of the trackers currents and the controller current (measured by the controller) it was considered beneficial to have independent sensors. These can be removed later if testing shows them to be no more accurate than the alternative.

The devices outlined in the following sections were designed to accommodate Sunswift's specific needs, based on the hardware and resources developed as infrastructure. Further additions to the system should be trivial requiring only production of the device as well as appropriate length cables for connection to the network.

8.1 Devices Designed

Along with the motor controller and MPPTs, a number of devices were designed to carry out the required functions within Sunswift. In-channel and out-channel IDs are defined in *scandal_devices.h*.

8.1.1 DC-DC converter

A DC-to-DC converter was constructed to act as a power supply for the low voltage system. It is connected to the high-voltage bus and generates the voltages required on the power rails of the CAN bus. The CAN bus is galvanically isolated from the high voltage supply via A VI-J00 isolated converter from Vicor which up to 75W at 12V. The 5V power rail is generated from this 12V using an efficient switching regulator.

Since it is connected to the high voltage bus, it is a good point to take battery voltage measurements. A CAN node (as specified by the reference schematics outlined in Chapter 4) is built onto the same PCB as the converter, with a high-precision 16-bit ADC providing accurate voltage samples. The voltage is oversampled (at approximately 1kHz) and digitally filtered in the microcontroller. The ADC is isolated from the microcontroller. Two measurements of the bus voltage are taken — before and after the fuse — such that the status of the input fuse can be determined. Current into the converter, and DC-DC chassis temperature are also measured. A connection for a cooling fan, if required, is available, and is switched using a MOSFET.

A number of values regarding the low voltage lines are measured by the microcontroller's internal ADC. A, switched, 12V output connector is also available.

8.1.2 RS232 card

In order to provide an interface for a PC or other device with a serial port, an RS232 daughterboard was designed for the CANRefNode. This board uses the ATMega323's internal UART and connects it to an RS232 line driver and the appropriate connector for communications with a PC's serial port.

Two pieces of software were built to run on this device. The first is a small terminal application which allows a user with a serial terminal to monitor the bus, send configuration messages, and read the error status of the network.

A second piece of software implements a serial CAN bridge. CAN packets are received using the controller, and sent to the serial port using a byte-stuffed protocol. Similarly, CAN packets are received via the serial port and protocol, and sent to the CAN controller. Thus any device with a serial port can effectively become a node in the CAN network.

8.1.3 Current sensor

A current sensor was designed to measure up to 50A using a hall-effect transducer. Since the transducer can produce both positive and negative voltages (corresponding to positive and negative currents) an isolated DC-DC converter was used to generate 0V, +15V, and +30V. The transducer was referenced to the +15V line, effectively shifting it by 15V. A resistor divider was used to scale the voltage from 0 through 30V to 0 through 3.0V (which the ADC can measure). Thus 1.5V is equivalent to 0A, 0V to -50A, and 3.0V to +50A. A second order butterworth LPF with a 1kHz cutoff was designed to reduce the noise at the ADC input. See the schematics in Appendix B for further details.

The 10-bit ADC on the microcontroller gives 0.1A per least significant bit change (since -50A to 50A is measured). This is probably not enough for high accuracy measurements (where 1mA would be preferable). This deficiency should be investigated in the future and remedied.

Current is sampled at approximately 1kHz and digitally filtered to provide an update every 0.1s.

The sensor can be shut down via a FET which controls the supply to the 30V DC-DC converter. Current used to supply the transducer is measured using a high side current sense resistor and associated amplifier.

The isolated ground is connected to the un-isolated ground on the daughtercard to simplify the design. This should not result in a decrease in reliability, since the node

never comes into physical contact with any external voltage because the transducer is completely isolated from the high-power lines.

8.1.4 Driver display

A 16x2 character *liquid crystal display* (LCD) was interfaced to a CANRefNode. A ribbon cable connects the node to the LCD screen. A FET can turn the LCD's backlight on. A potentiometer adjusts the contrast voltage. A suitable LCD screen, which could accept 3.3V as a supply voltage, was found, meaning that no voltage translation was required for control lines (as would have been had a 5V LCD been used).

Following an unsuccessful attempt to write the software required to drive the LCD, a public-licence library was found which worked extremely well. The display has been programmed to accept text messages to be displayed on the screen as well as values, via in-channels. Four numbers are displayed, the displayed values are configurable by changing the in-channel source.

Other displays could be used — larger character displays would be particularly suitable such that more telemetry channels could be shown.

8.1.5 Handlebar Controls

The driver requires a number of controls to operate the car. Most notably a throttle control. Having two configurable buttons was also considered useful. The Tritium throttle control board is not suitable for mounting on the Sunswift handlebars, and therefore one was designed. A PCB with two under-thumb buttons was designed. These controls have been mounted neatly on the right hand handlebars.

8.1.6 Driver controls interface

In order to interface the switches and buttons present on the handlebars, a driver controls interface board was designed to interface to the CANRefNode. The board is very simple, consisting only of connectors wired to the microcontroller's pins. The ATMega323 has inbuilt pull-up resistors. This interface should be rebuilt with added protection circuitry for the inputs. An excellent scheme was outlined by a Tritium electronics engineer. The board connects to the small handlebar mounted switchblock used in Sunswift II, the thumb-buttons board outlined above, and a rotary encoder.

The software for this board provides out-channels corresponding to the left indicator, right indicator, horn, and display scroll up/down. Messages are sent to turn the indicators on and off once per second when activated. This means that the software on the nodes controlling the indicators can be identical to the software controlling battery fans, the horn, etc. This allows for easy replacement of a faulty node, as well as decreased complexity.

8.1.7 Switch card

A second year student, Bonne Eggleston, designed, with the help of the author, a board to supply power at 12V to whatever devices should require it. The board uses a P type MOSFET to switch the high side of the device on and off. Each channel has current sensing, as well as pre and post-fuse voltage sensing (allowing for blown fuse detection). Up to 3A can be supplied from each channel.

The software written accepts two in-channels indicating the state of each connector. When a value with a more recent time stamp is received, the output is updated depending on whether the in-channel is zero or non-zero.

8.1.8 1-Wire temperature sensors

The same student also designed an interface to the 1-Wire network, in order to read a number of temperature sensors. A UART to 1-Wire converter chip was used to achieve this interface.

The node allows a network of low cost temperature sensors to be connected using only two wires. These temperature sensors are then read and the values transmitted over CAN using individual channels.

The software written is based on the 1-Wire public domain kit. This software requires UART send and receive functions, and these are provided using the abstraction library. The high level software on the node was written by the student, using the libraries outlined in Chapter 5.

8.1.9 Palm pilot

An m100 Palm Pilot was seen as the cheapest way to service the car with a graphical LCD. It was seen as important that the drivers could observe their power usage patterns. The LCD should be able to display graphs and other representative information regarding any of the parameters within the car.

The Palm is to accept data via its serial port, from an RS232 node (as described in Section 8.1.2), using the byte-stuffed protocol which has been implemented. An SRT student was given the task of writing the software to perform this function, but has, as yet, made only preliminary progress. The Palm's power is supplied from an internal battery.

8.1.10 PLEB

The PLEB, developed by CSE, is intended as a more powerful processing node in the network. It currently runs the Linux operating system. Software has been written to act as a transparent CAN bridge, taking data received via the serial port, and transmitting it to a support car using the wireless Ethernet protocol.

Harvey Tuch, a CSE undergrad student, assisted in the setup of the PLEB. Catapult, the bootloader under development by the distributed systems group at UNSW, was not mature enough to be used for the application, and therefore BLoB, developed by the LART team at TUDelft, was used in this role. Unfortunately, BLoB's flash memory writing routines are not currently compatible with the PLEB hardware, and therefore a workaround was used involving a network booted Linux kernel which had flash memory writing drivers installed.

The PLEB is configured with the ssh daemon available for on-road maintenance. An IP assignment system was instituted for addressing the solar car and support car.

8.1.11 Miscellaneous devices

Two other devices were developed:

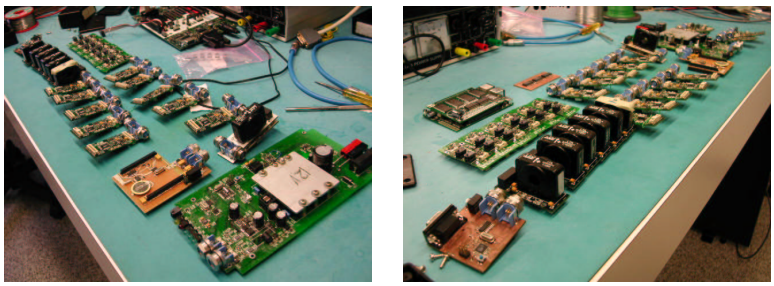


Figure 8.1: Approximately half of the PCBs constructed for Sunswift

- an IV curve sweeping board which can measure the IV characteristics of a solar panel several times per second
- a power supply for testing and configuration of the network without the batteries

8.2 Construction

The facilities at CSE are appropriate for building prototype boards, but are costly, and time consuming, for production runs. In order to produce the multitude of PCBs that were required, two external manufacturers were employed. IMP Printed Circuits provide a cheap service for the production of detailed and multi-layer PCBs. The CANRefNode was designed with this business's service specifications, and used tracks and spacing down to 5 thousandths of an inch (0.13mm). BEC Manufacturing was used for a second run of PCBs which did not need to be so detailed because of their prototyping service where a number of PCBs can be placed on a single panel for a fixed (low) cost.

Components were ordered by Sunswift's purchasing officers. While Farnell components were chosen, a 60% reduction in price could occasionally be achieved by ordering from a different supplier. Farnell was used as a fall-back should other sources not be available.

The boards were populated by first year and second year students from the Sunswift team under the guidance of the author, and David Johnson — a senior technical officer from CSE. The *surface mount technology* (SMT) lab at CSE was used for this purpose, providing quality tools and viewing systems. David Johnson checked the PCBs for soldering errors, and provided fixes where necessary.

Testing of each board for function was performed by SRT students, and malfunctioning boards were given to the author to be repaired.

In total, over 50 devices were manufactured by six students.

8.3 Installation

The above outlined devices were installed in the car in various stages. Until the newly developed system had the same, or better, functionality than the old electrical system, the two were run in parallel. Following the gaining of equivalent functionality, the old electrical system was removed from the car, and the new system permanently installed.

Wires to connect the CAN network nodes were constructed by the Sunswift students using category 5 twisted pair cable and the Triad connectors. Molex Microfit connectors were used to make miscellaneous connections to the nodes.

The DC-DC converter is connected to the high voltage switchblock via a fuse, and is located behind the driver on the left hand side of the car.

Switchcards were used to service the indicators, with one at the front servicing the left and right, and two at the back servicing the left and right indicators and the brake lights. Two switchcards are connected to four battery cooling fans. Another is used to service the horn and rear-vision screen.

Two RS232 devices are used: one connects to a Palm Pilot, and another connects to the PLEB. Both use the byte-stuffed binary protocol over RS232. The PLEB is installed behind the driver's seat on the right. The wireless Ethernet transceiver is also located behind the driver's seat, on the bottom shell.

A driver controls board was installed, and the necessary wiring/crimping done to connect the handlebar controls to it.

Three current sensors are used to measure the battery, array and motor controller currents.

A driver display board was mounted in the centre of the array, and the character LCD situated just under the visor. This makes the display easy to find and read when driving the solar car. The palm pilot was mounted in the right hand control box by an SRT member. While this location is not within the driver's peripheral vision, it is thought that any information displayed is not likely to be critical.

The MPPTs are connected to the network in the top shell, again using category 5 cable. The wires were terminated using bootlace ferals, and screwed into the tracker's CAN connector.

The Tritium motor controller was connected via the adapter board outlined in Section 7.2. Other controllers were not considered since the software for the necessary adapter boards had not yet been written. The Tritium driver controls board is situated in the right hand control box.

Each end of the CAN bus is terminated using a 100Ω resistor.

Special attention was paid to the order in which the devices were wired. The most important devices were located close to the DC-DC converter, so that a break in the power supply for these nodes is more unlikely. The motor controller and PLEB are on either side of the converter.

Configuration of the system occurred as new devices became available. Calibration of the scaling constants was completed to the point where each device was useful, but not further. Interesting, but non-essential channels such as the current flowing from the DC-DC converter were not calibrated. Critical channels such as the battery current sense were carefully calibrated.

8.4 Testing and Debugging

The system was tested progressively as it evolved. All components were bench-tested as they were developed, but bench conditions are not a good test for the robustness of the hardware. Vibration analysis and simulation was considered by one member of the team, but eventually it was decided no substitute was available for on-road testing. Six testing days were used to test the system, with a rigorous testing schedule planned during the summer break.

8.4.1 Testing day 1

Two prototype current sensors, a DC-DC converter, and the MPPTs were used to monitor the car on a trip along the M5 motorway. The old electrical system and driver controls remained. An RS232 card in conjunction with a Proxim wireless radio modem was used to communicate with a laptop computer. The system was driven, and the data logged using a small application running under Linux. Only some data were obtained from the MPPTs because most of them had not yet been modified to be compatible with the Sunswift bus arrangement (particularly the power supply voltage levels).

The testing run proved that the network operated in its basic form. The most debilitating bug was the accidental inclusion of debugging code in the current sensors. Whenever a message send failed, the current sensors would enter an error mode, flashing the indicator LEDs to indicate which error occurred. Unfortunately, this also meant that no messages were sent by the sensor in this mode. The system would, therefore, operate for a short period before the current sensors would disable themselves. In order to avoid this in the future, the in-car software should be frozen several days before any race occurs such that testing may discover any obvious faults such as these.

8.4.2 Testing day 2

Testing day 1 was the only day where the two electrical systems were run in parallel. Between testing day 1 and testing day 2, the car's electrical system was entirely replaced, and the Tritium motor controller installed (along with the driver controls and control boards associated with it). The car was again driven along the M5 motor way until the Tritium controller failed because of a current limit setting that stopped it climbing a steep hill.

For testing day 2, the driver display, indicators, motor controller and driver controls were added to the CAN network. Speed, battery voltage, array current, and motor controller current were displayed, and worked well. The indicators seemed to operate poorly, flashing at a non-uniform rate. This bug is still not quite worked out, but one problem was associated with the code on the driver controls rather than that at the switch cards. The issues are also related to the MCP2510's garbled message issues. The problem appears to get worse with increasing bus traffic, which corresponds with the problems set out in the MCP2510's errata sheet.

No wireless communication was available for this testing day. The driver display was used to monitor the car from within.

8.4.3 Testing day 3

In preparation for a race which was later cancelled, the team built a chain drive system such that the car would be able to climb steep hills. In order to test this, the car was driven up the Mount Victoria pass (which is on the opposite side of the Blue Mountains to Sydney). Again, no wireless communication was available, and the car was monitored using the driver display, current sensors and DC-DC.

8.4.4 Testing days 4 & 5

During the mid-session break, the car was taken for a weekend drive: first down the M5 motorway, then turning towards Cowra.

The PLEB and associated serial card and the wireless Ethernet radio link were installed for the testing run. Scandalconf was used to log the data.

Unfortunately this run was marred by the malfunction of the Tritium motor controller. On the first day, bad current control loop constants had been programmed into the controller, and it was swapped for the UQM (meaning that no telemetry data were obtained, and no speedometer was available). On the second day, following a very short period on the road, the Tritium controller damaged itself when its power supply, which had not been secured sufficiently, became disconnected. Note that the rest of the network survived, despite the destruction, due to the isolation barrier afforded by the adapter board built for the Tritium controller.

The indicator problem still had not been addressed.

Tracker communication was unreliable. It was later discovered that the supply voltage for the trackers was dropping. This was because the trackers require exactly 5V for their CAN communications circuitry (as opposed to the >3.3V required by the CANRefNode and Tritium adapter board). When this supply voltage dips, the tracker's communications circuitry fails and sends spurious messages on the network — effectively clogging it. This was solved for the last testing run by adding a 5V LDO regulator in the at the end of the bus. The cause of this voltage drop should be further investigated.

Two way communication with the PLEB was not possible due to a bug in the wireless Ethernet protocol code. This has not yet been resolved, but is not critical since its main function is to supply the support car with telemetry data. When control of the car is required, two-way communication will become important.

8.4.5 Testing day 6

The most recent testing run was held on 27th October. Two and a half hours were spent driving the car. Apart from the absence of temperature monitoring, and the exclusion of the horn due to a lack of available working CANRefNodes, the car was in its intended state. Software wise, the two-way communications had not been resolved, the CAN network still appeared to occasionally garble messages, and the Palm Pilot display software had not been written. Apart from these faults, the telemetry and control system functioned well.

The car was forced to stop on three occasions because of a failure in the Tritium motor controller. This was due, again, to a malfunction of the current limiting code. As long as the current was kept under the limit, the controller functioned well. The telemetry system was used to guide the driver as to how fast to go in order not to exceed this current[1] (which was viewable as a telemetry channel in the support car). The telemetry data, in this instance, have saved a significant amount of time at the side of the road, and would have been invaluable in a race situation. Had two way communications been running, the driver display could have been reconfigured to display the motor current while still driving.

[1] Note that this current is the *motor* current rather than the motor *controller* current.

Figure 8.2: The author monitors the solar car during testing day 6

Chapter 9

Results

A number of different results can be drawn upon in the evaluation of the development undertaken during this project.

9.1 Sunswift Requirements Fulfilment

In Section 3.1, requirements for a solar car's electrical system were outlined. The system developed (using the infrastructure also developed) fulfils those requirements in the following ways[1]:

- *allow the collection of data gathered in the operation of devices within the car*. Both the Tritium motor controller and Biel maximum power point tracker have been successfully interfaced to the control network within the car. These devices have been used to gather telemetry data;
- *provide status information on a device's functioning, and mode of function.* Each device provides a heartbeat message once per second with useful diagnostic information (such as most recent error). This information is not currently analysed by Scandalconf, but this is simply a matter of implementation;
- *provide feedback to the car's driver.* A driver display has been installed and used effectively;
- *perform well in an electrically noisy environment.* Analysis of the errors which occur on the bus indicate that errors are unlikely in the steady state (e.g. when driving). The bus does not appear to be adversely affected by EMI;
- *provide facilities for control of devices within the car.* Control of the devices within the car has been implemented and works well, but not using the wireless link afforded by the PLEB. This is due to a software error and should be resolved. Battery fans can be controlled, and messages sent to the driver display;
- *be fault tolerant. One component's malfunction should be isolated to the failure of that component.* This has been proven via the system's survival of various malfunctions. The Tritium controller's malfunction during testing day 5 had

[1] The comparisons given here are only those which it was felt required justification or have not been fulfilled. Requirements which have obviously been fulfilled are omitted.

no effect on the rest of the system (which continued to operate). Furthermore, with the replacement of the motor controller (and only the motor controller) the system continued to operate;

- *use a minimum of energy.* The current electrical system uses approximately the same power as the previous one (≈10W). Note that power consumption could be reduced significantly by the improvement of the DC-DC converter. It could also be reduced significantly via the use of un-isolated nodes for non-critical systems.

9.2 Comparison with Previous System

The system implemented using infrastructure developed is superior to the previously used system in that it fulfils the requirements for a solar car's electrical system in a more useful, more reliable manner. The success of the new system is only possible because of the underlying control network, and the design decisions associated with its implementation.

9.2.1 Extendibility

The system is inherently extendible, as was demonstrated through Sunswift's electrical system implementation. First only four nodes were present, monitoring three currents and a voltage. Later on, nodes were added to operate switched accessories, a driver display was added as well as the PLEB, Palm Pilot, etc. Additions still planned include array and battery temperature sensor networks, further driver displays, tyre pressure sensors, a battery monitoring system, etc. The design of these nodes is abstracted by the network and Scandal protocol.

In comparison, take the team's attempt to add a driver display to the previous electrical system in 2001. An LCD screen with an RS232 interface (which increased the cost of the device significantly) was purchased, however all of the available RS232 ports on the uCsimm were in use. Thus it was necessary to add one. In order to do so, several ICs would have been required to be soldered onto the prototyping board to which the uCsimm mounts, or a new carrier board for the uCsimm designed. Both options would have taken a significant amount of time. Furthermore, there were nine other devices in the car possessing serial ports. Nine more UARTs would have to be added. Compare with the same exercise using the newly developed infrastructure: the RS232 daughtercard could be connected to a CANRefNode, and software written.

9.2.2 Flexibility

The system is inherently more flexible than the previous one. Take the example of the various switches on the handlebars. These switches can be reconfigured to be used for many purposes. The right hand buttons could be configured to turn the battery fans on and off, rather than scroll through the screen, simply by configuring the in-channels in the screen and battery fans differently. For example, should the battery fans be required to switch on when the brakes are active (for whatever bizarre reason), it would be a simple matter to write the software to accomplish this. Attaining the same functionality in the old electrical system would have been extremely difficult — requiring the addition of hardware.

	Old system	New system
Number of telemetry channels:	11	153
Minimum sampling period:	5 s	0.01ms
Maximum sensor-measurement distance:	2m	5cm
Tracker and controller status?:	No	Yes
Number of external sensors:	3	3

Table 9.1: Comparison of obtained telemetry to the Sunswift II, 2001 telemetry system

9.2.3 Data quantity and quality

The new electrical system can currently log up to approximately 400 packets per second, and has a theoretical limit of 8000 packets per second. With this data transfer capability, the rate at which data can be logged is greatly improved.

The number of telemetry channels is significantly increased. See Section 9.3 and Appendix C.

9.3 Test Drive Logs

On the test drive outlined in Section 8.4.5, data were logged using Scandalconf. The following graphs have been generated from those logs (and could have been generated in real-time were the software written). This demonstrates the kind of information, gathered by the new electrical system, that can help make informed strategy decisions, as well as verify the correct operation of the car.

Figure 6.13 was generated from data taken on testing day 6. It shows the power generated by each tracker as calculated using the panel voltage and current information that is transmitted via the network. The break at 4550 seconds is due to a fuse failure at Pheasants Nest fuel station which disabled the receiving wireless Ethernet access point.

Figure 9.1 is interesting because it shows an important ability of the telemetry system: estimating the battery state of charge. As can be seen, BSOC is depleted as the batteries provide energy.

Figure 9.2 shows the power into the motor controller vs. the speed of the car. Unfortunately, the M5 has very few level sections. For this reason, it is difficult to estimate the car's CdA and C_{rr} from this data. A curve calculated using estimated parameters is overlayed in the same figure.

9.4 Battery Discharge Monitoring

There were two motivations for monitoring controlled discharges of the Sunswift battery pack. The first was to use the data gained to estimate the state of charge at a given voltage, since for Li-ION batteries, there is a strong relationship between the battery state of charge and the cell voltage. The second was to demonstrate the flexibility of the framework constructed — by using the same technology in a slightly different application.

A load consisting of a number of electric oven elements was used to dissipate energy as heat when discharging the pack. A carbon pile was used as a variable resistor

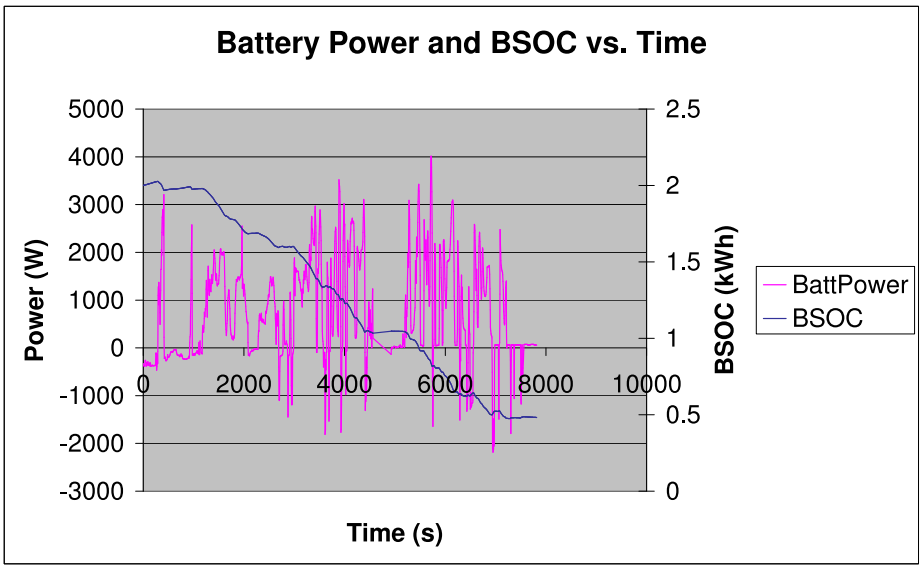


Figure 9.1: Battery state of charge vs Time for testing day 6

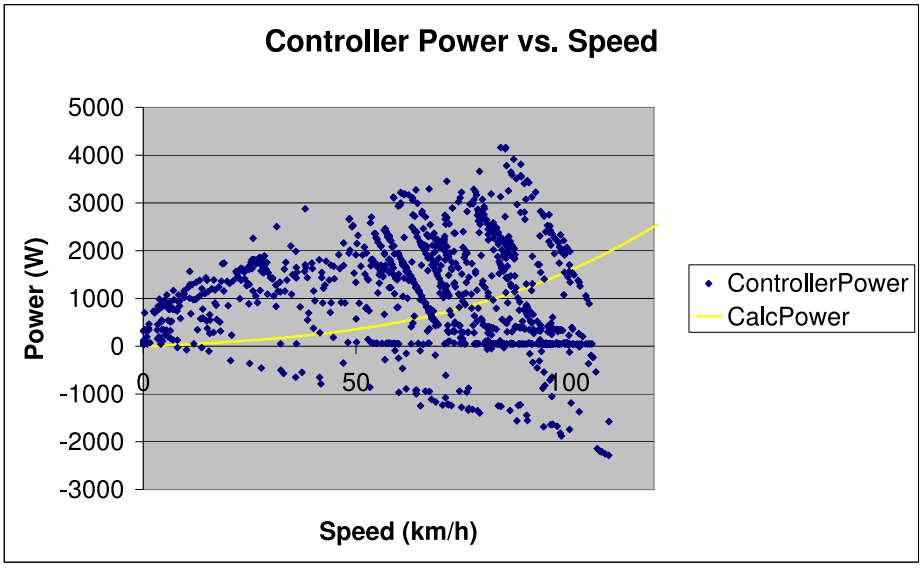


Figure 9.2: Motor controller input power vs speed for testing day 6

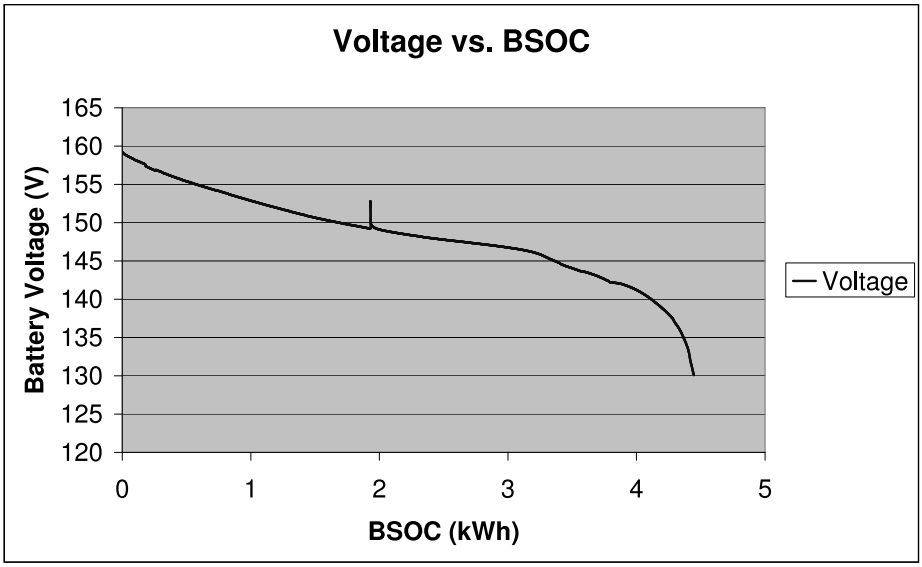


Figure 9.3: Voltage vs battery state of charge for Sunswift II WSC battery

in order to adjust the current at which the pack discharged. The pack was discharged at 7A from 164V to 130V.

The discharge was recorded using a current sensor, two temperature sensors, a DC-DC converter (to measure the pack voltage) and an RS232 board. The data were logged on a PC using a textual version of Scandalconf. The driver display was also used to display the voltage, current, and battery temperatures.

The set up time for the monitoring system, excluding the time taken for some bug-fixes to Scandalconf, totalled approximately 10 minutes.

The data obtained were processed to determine the relative BSOC at each point of the discharge. The voltage was then plotted against it. This is shown in Figure 9.3. The spike in the middle of the discharge curve is where it was temporarily stopped while the people observing the battery observers left the building (and since no current was being drawn, the voltage rose).

Chapter 10

Conclusion and Future Work

Sunswift II's ad-hoc, inherently limited, telemetry and control systems have been replaced by a system *designed* from the ground up to be integrated, extendable, flexible, and most importantly, reliable — to form a *system* rather than a collection of devices.

This re-development has involved the design and implementation of the hardware required to provide the solar car with the necessary functionality. Furthermore, the software which is required to run on the hardware developed, has been written, and debugged, such that a working system has resulted.

One of the most important pieces of this development was the car's maximum power point tracker software. The devices have been significantly improved through this modification, as has been demonstrated via the 2nd placing in Sunrace, 2002. The reliability of this system compared to the same device with the World Solar Challenge code, in that race, is vastly superior. No MPPT failures have occurred since the software's commission, compared with nearly 50% failure in the period of one week during the WSC.

The car has been driven on numerous occasions in both racing and race-simulating conditions at various stages of the new system's development. A number of successes and a number of failures have occurred. The electrical system has proven to be progressively more reliable as testing uncovers new bugs, as well as new solutions to these problems.

One of the most important results is that following this work, the team has had experience with the internal workings of all of its primary components, including the MPPTs and motor controller. Should repairs be necessary, or improvements be possible, the team is likely to be able to take advantage of this knowledge.

Some of the individual achievements and conclusions have been outlined in the following sections.

10.1 Hardware and Software Infrastructure

The CANRefNode and associated Scandal protocol form the bulk of a framework developed in the course of this project. The combination of these two has been shown (through the development of Sunswift's electrical system) to be a powerful prototyping and development environment. Useful, customised embedded systems and embedded networks can be constructed in a limited time, and, more importantly for the UNSW SRT, modifications can be made in similarly short periods.

The resulting system is designed specifically for Sunswift, but is, at the same time, based on a generic, powerful framework. As such it has the advantages of custom design (reduced size, reduced weight, the appropriate connectors, etc).

10.2 Biel MPPT

Another important part of this thesis was revised operating software for the Biel MPPT. The device's reliability, with the addition of this software, has been significantly improved. A simple open-loop MPP tracking algorithm has been implemented and tested. The Scandal protocol has been built into the software in order to provide a network with telemetry information. A number of other useful features have been implemented including in-tracker IV curve sweeps, and serial based telemetry. The tracker code is difficult to develop further due to code size limitations, but significant improvements can be made either through the reconstruction of the control board, integrating improvements in the areas identified (giving more program memory), or through the optimisation of the existing software.

10.3 Tritium Motor Controller

The Scandal protocol has been integrated into the Tritium motor controller code, making it compatible with the other devices developed. The resulting network node has been tested on-road and has performed well. Motor controller failures during testing were not related to the operation of the new code or the protocol. Many of the failures have been diagnosed via the use of the new network.

10.4 Future work

Future work on the framework would largely depend on the application for which it is being developed. However:

- The microcontroller and CAN controller used on the CANRefNode should be investigated further, and more appropriate alternatives used if found. The MCP2510, in particular, is difficult to use, and its replacement should be considered for new designs. The microcontroller chosen uses more power than alternatives.
- An un-isolated version of the CANRefNode should be designed for use where isolation is not beneficial. An example of this is the current-sensor board, which bypasses the isolation used. This would reduce both cost and power consumption of the devices.
- A method is required to read the configuration from a CAN node and save it.
- Scandalconf is only considered a preliminary tool, and should be either rewritten or substantially revised such that it fulfils the desired functions. A number of useful user interface features could be implemented in such an application.
- A bootloader should be written which allows nodes to be reprogrammed over the CAN network. This would save a substantial amount of time when applying software revisions to a network of nodes.

Future work to the motor controllers should include:

- further bench testing of the remote-control of the set-speed used by the speed control loop within the Tritium controller. This should then be tested in the solar car;
- software being written for the adapter boards such that Sunswift's less modern, but proven, controllers can be operated using the same interface as the Tritium.

Work recommended regarding the solar car:

- Further investigation should be carried out regarding the CAN code. Issues to resolve include those associated with the MCP2510 errata, and real-time issues associated with slow processing of CAN data. This should fix the problems currently being experienced with the indicators.
- A comprehensive battery monitoring system should be investigated. Li-ION batteries require constant monitoring normally provided manually via . In order to satisfy this need, a cell-level
- The current sensors which have been implemented have a precision of only 0.1A. This is deemed too low for reliable current measurement. Furthermore, the transducers used exhibit an odd hysteresis like effect around zero current. The sensors should be investigated and rebuilt if necessary. One alternative might be to use current-sense shunts rather than hall-effect transducers for current measurement. A higher resolution ADC should almost certainly be used.
- Temperature sensors should be installed in all four battery packs and the solar array.
- The PLEB should be directly connected to the CAN network. While a daughtercard to connect the PLEB to a CAN network was designed and built, incompatibilities with the available Ethernet card eliminated the possibility of its use. This situation should be investigated. It is recommended that a more powerful CAN controller be selected for this purpose.
- Some form of writable mass storage should be implemented for use with the PLEB. This would allow easier software updates, self hosting, and data logging capabilities. If data were logged within the car, only critical information would need to be sent via the wireless link, the rest being stored for later retrieval.
- Software for the Palm Pilot should be implemented such that it can act as an effective graphical driver display.
- Rigorous testing of the car's electrical system should be undertaken to ensure race reliability. A testing program should be developed. A routine cycle of driving the car, repairing faults found, and re-testing should be instigated.
- A thorough maintenance checklist be developed for use on a nightly basis during races to decrease the chance of in-race failure.

Suggested future work in relation to the MPPT software is outlined in Section 6.7.

10.5 Benefits for external parties

It is believed that this development could be used not only for the solar car, but in a number of other applications. It is trivial to adapt the CANRefNode to perform a number of tasks using custom-built daughtercards, but even this would not be necessary for a great number of applications given the daughtercards already available.

One engineering student working on the bio-diesel project with the Centre for Photovoltaic Engineering has expressed interest in implementing a more advanced monitoring and control system. This project aims to build a trailer with a small plant capable of producing a diesel replacement using kitchen cooking oil. This should prove relatively straight-forward, with almost all of the hardware and software components already built and tested. Some further hardware to handle the high currents associated with large motors used for pumps and stirrers would be required, as well as some user-interface software. Components such as the character display, temperature sensors, switch cards, and serial boards would be easily integrable. Some aspects of the CANRefNode, such as the isolation barrier and connector used, might be reconsidered in order to reduce cost in this application where these features are of little advantage.

Another organisation which may benefit from this work is the UNSW satellite project, who are currently attempting to design and build a flight computer for BLUEsat, a small micro-satellite to be launched in 2004. Some kind of distributed system makes a great deal of sense from a reliability point of view (in the same way as it does for a solar car). The technical leaders working on the project have expressed a concern regarding the power such a system might consume, but it is thought that this could be reduced to a very low level via the use of microcontroller's sleep modes. With such a small satellite, CAN may not be the most appropriate technology to employ, but many of the lessons learnt can be transposed.

As another completely unrelated application, the author intends to develop a Scandal based theatrical lighting system.

The environment developed in this thesis is not intended to provide an off-the-shelf solution. But *is* designed to be an easy-to-use prototyping and development platform for embedded networks and embedded systems. In this way it is well suited for use by other student projects within the university for a variety of purposes, as well as external parties with similar needs.

10.6 Final Words

Solar car races like the World Solar Challenge are intended to spark continuing development of renewable energy technologies and energy-efficient vehicle design. Significant pieces of technology such as the world record holding PERL solar cells and highly efficient in-wheel motors have been developed as a result of these races. As a participant, the UNSW team should continue to research new and innovative ways of improving their car, with the goal of improving the overall technology in a significant way. This project has given the team a base unifying framework with which to build and evaluate their innovations. The most important idea to understand is that any innovation must be reliable, and even a straight-forward modification can not be considered reliable until it has been well tested.

Bibliography

[1] Solar Motions team. http://www.solarmotions.com, 2001.

[2] World Solar Challenge race regulations. http://www.wsc.org.au/Rules/regulations-text.htm, 2001.

[3] World Solar Challenge results. http://www.wsc.org.au/Results/2001/final.solar, 2001.

[4] Farnell home page. http://www.farnell.com, 2002.

[5] Open DeviceNet Association. Open DeviceNet Association home page. http://www.odva.org, 2002.

[6] BRUSA Electronik AG. MPT-6 datasheet. http://www.brusa.biz/products/e_mpt_6206.htm, 2002.

[7] Atmel Corporation. AVR910: In-system programming. http://www.atmel.com/atmel/acrobat/doc0943.pdf, November 2000.

[8] Atmel Corporation. ATMega323L datasheet. http://www.atmel.com/atmel/acrobat/doc1457.pdf, September 2001.

[9] Fluke Corporation. Fluke hydra data logger. http://www.fluke.com/products/home.asp?SID=7&AGID=6&PID=5308, 2002.

[10] New Generation Motors Corporation. NGM website. http://www.ngmcorp.com, 2001.

[11] Jeffrey E. Cotter, David M. Roche, John W.V. Storey, Antony E.T. Schinckel, and Clive P. Humphris. *The Speed of Light 2*. Key Centre for Photovoltaic Engineering, University of New South Wales, 2001.

[12] Paul Craven. Solar car power management. Master's thesis, University of Missouri-Rolla, 1996.

[13] Lars-Berno Fredriksson. On the difference between CANOpen and CanKingdom. Technical report, Kvaser, AB.

[14] Lars-Berno Fredriksson. CanKingdom specification. http://www.cankingdom.org/download/Downloadfiles/CanKingdom/ck301.zip, 1996. Version 3.01.

[15] Thomas Fuhrer, Bernd Muller, Werner Dieterle, Florian Hartwich, Robert Hugel, and Michael Walther. Time triggered communication on can. In *7th international CAN Conference*. Robert Bosch GmbH, 2000.

[16] Robert Bosch GmbH. CAN specification 2.0B. `http://www.can.bosch.com/docu/can2spec.pdf`, 1991.

[17] D.P. Hohm and M.E. Ropp. Comparative study of maximum power point tracking algorithms using an experimental, programmable, maximum power point tracking test bed. In *Conference Record of the Twenty-Eighth Photovoltaic Specialists Conference*, pages 1699–1702, 2000.

[18] K H Hussein, I Muta, T Hoshino, and M Osakada. Maximum photovoltaic power tracking: an algorithm for rapidly changing atmospheric conditions. *IEE Proceedings- Generation, Transmission and Distribution*, 142(1):59–64, January 1995.

[19] CAN in Automation. CAN in automation web site. `http://www.can-cia.com`, 2002.

[20] CAN in Automation. CANopen product guide. `http://www.can-cia.com/products`, 2002.

[21] CAN in Automation Users and Manufacturers Group. Canopen specification, v3.0, October 1996.

[22] Uwe Koppe. CANpie user manual. `http://www.microcontrol.net/CANpie/cp_download.php3`, October 2002.

[23] Kvaser. Kvaser home page. `http://www.kvaser.com`, 2002.

[24] G. Leen and D. Heffernan. Expanding automotive electronic systems. *IEEE Computer*, 35(1):88–89, January 2002.

[25] Kent Lennartsson and Lars-Berno Fredriksson. Fundamental parts in SDS, DeviceNet and CanKingdom. `http://www.kvaser.com/can/hlps/devsdsck.htm`, December 1996.

[26] Paul N. Leroux. Redundant network links boost reliability. *Embedded Systems Engineering*, 2001.

[27] Jochen Liedtke. On μ-kernel construction. In *sosp95*, pages 237–250, Copper Mountain, CO, USA, December 1995.

[28] Surrey Sattelite Technology Ltd. SSTL modular microsatelite. `http://www.sstl.co.uk/datasheets/Microsat.pdf`, 2002.

[29] Tritium Pty. Ltd. Tritium intelligent vehicle controller. `http://www.tritium.com.au/Products/ProductsMotorController.htm`, 2001.

[30] Microchip Technology Inc. *MCP2510 Errata*, 2001.

[31] MoTeC Australia. MoTeC advanced dash logger. `http://www.motec.com.au/adl.htm`, 2001.

[32] HMS Industrial Networks. HMS Industrial Networks homepage. `http://www.hms.se/`, 2002.

[33] Nonvolatile Electronics, Inc. The IsoLoop advantage. `http://www.nve.com/isopdf/advantage.pdf`, 2002.

[34] Biel School of Engineering and Architecture. MPPT new generation users manual. `http://www.hta-bi.bfh.ch/E/induel/r_and_d/MPPTnG/Tech.html`, February 2001.

[35] Biel School of Engineering and Architecture. New generation trackers. `http://www.hta-bi.bfh.ch/E/induel/r_and_d/MPPTnG/MPPTnG.html`, 2001.

[36] Maxim Integrated Products. Maxim 1-wire and ibutton website. `http://www.maxim-ic.com/1-Wire.cfm`, 2002.

[37] Proxim Inc. Proxim EA-7291 RangeLAN2 Ethernet adapter datasheet. `http://www.powercorp.com.au/PDFs/B_7921.pdf`, April 1998.

[38] Peter Pudney. *Optimal energy management for solar-powered cars*. PhD thesis, University of South Australia, `http://scg.levels.unisa.edu.au/Pudney/thesis`, August 2000.

[39] V S Ramsden, B C Mecrow, and H C Lovatt. Design of an in-wheel motor for a solar-powered electric vehicle. *IEE EMD*, 1997.

[40] D.M. Roche, A.E.T. Schinckel, J.W.V. Storey, C.P. Humphris, and M.R. Guelden. *Speed of Light*. Photovoltaics Special Research Centre, UNSW, 1997. ISBN 0-7334-1527-X.

[41] Philips Semiconductors. The I^2C-bus specification, January 2000.

[42] D. Snowdon, J. Green, P. Cousins, S. Stone, R. Simpson, and J. Cotter. Composite curved laminates for the UNSW Sunswift II solar array. In *ISES Congress Proceedings*, 2001.

[43] IEEE Computer Society. IEEE standard 802-2001, IEEE standard for local and metropolitan area networks: overview and architecture, March 2002.

[44] Goro Tamai. *The Leading Edge*. Bentley Publishers.

[45] Alpha Centauri Team. Alpha Centauri solar car. `http://www.alpha-centauri.nl`, 2001.

[46] Iowa State University Team PrISUm. ISU Bitbucket. `http://www.drgw.net/workshop/solarcar/solarcar.html`, 1999.

[47] Thomas Toth, Robert Reid, and David Snowdon. Sunswift low voltage specification. Technical report, Univeristiy of New South Wales Solar Racing Team, 2001.

[48] International Telecommunications Union. Open systems interconnection — basic reference model: The basic model, July 1994.

[49] Adam Wiggins. PLEB: a platform for portable and embedded systems research. Technical Report UNSW-CSE-TR-105, School of Computer Science and Engineering, University of New South Wales, 2001.

Appendix A

CANRefNode Schematics and Artwork

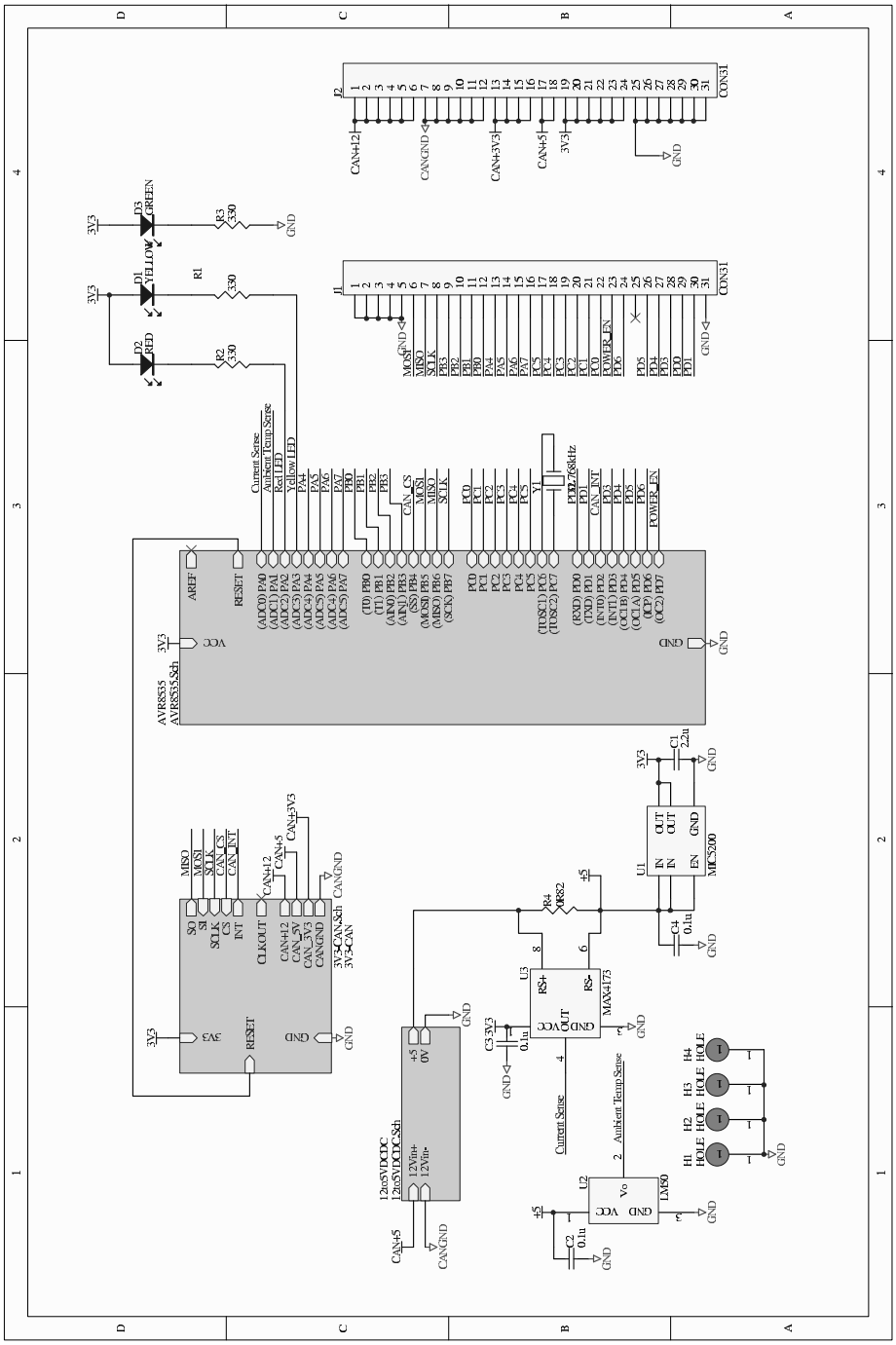


Figure A.1: CANRefNode top level schematic

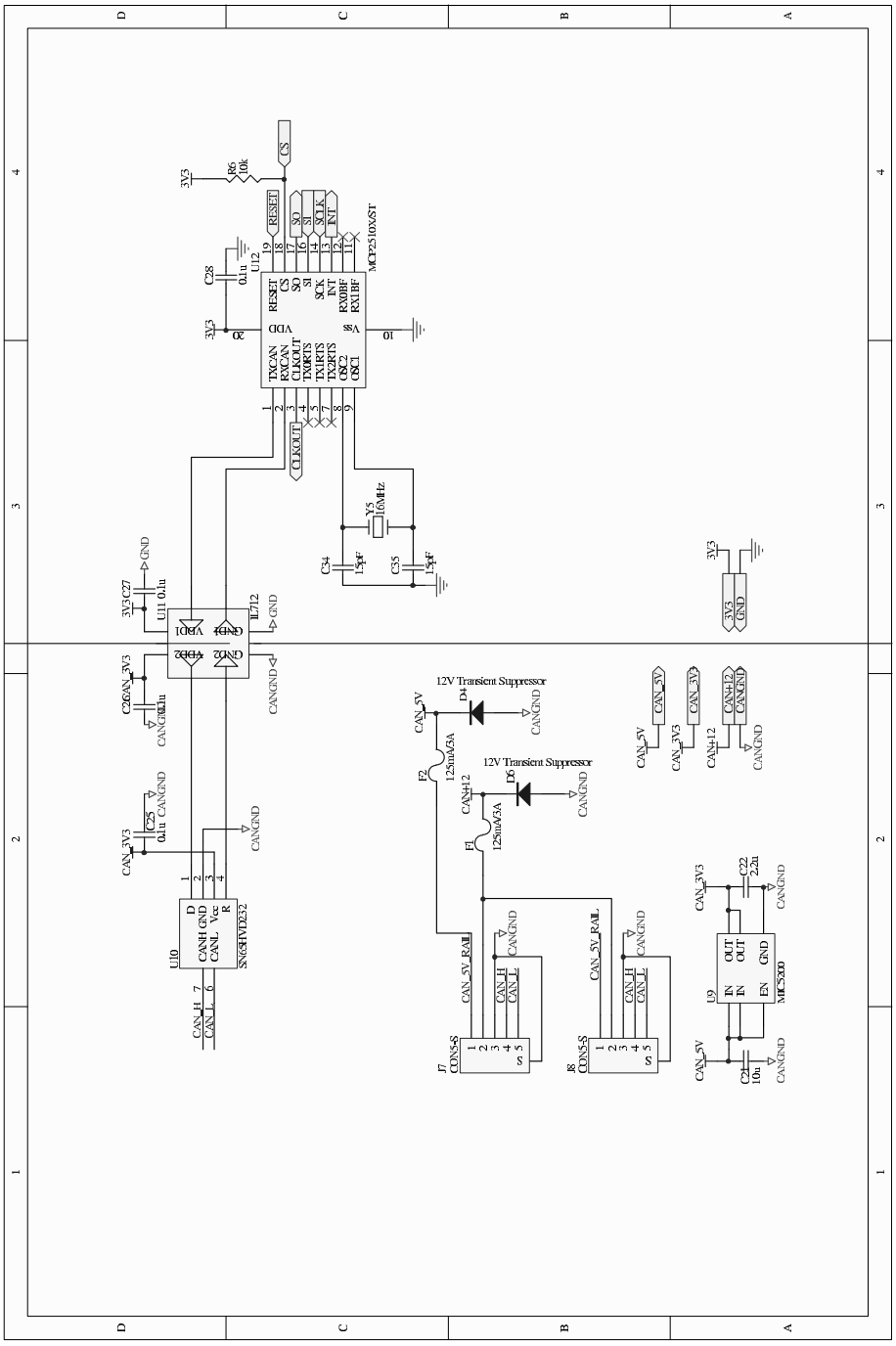


Figure A.2: 3.3V CAN module schematic

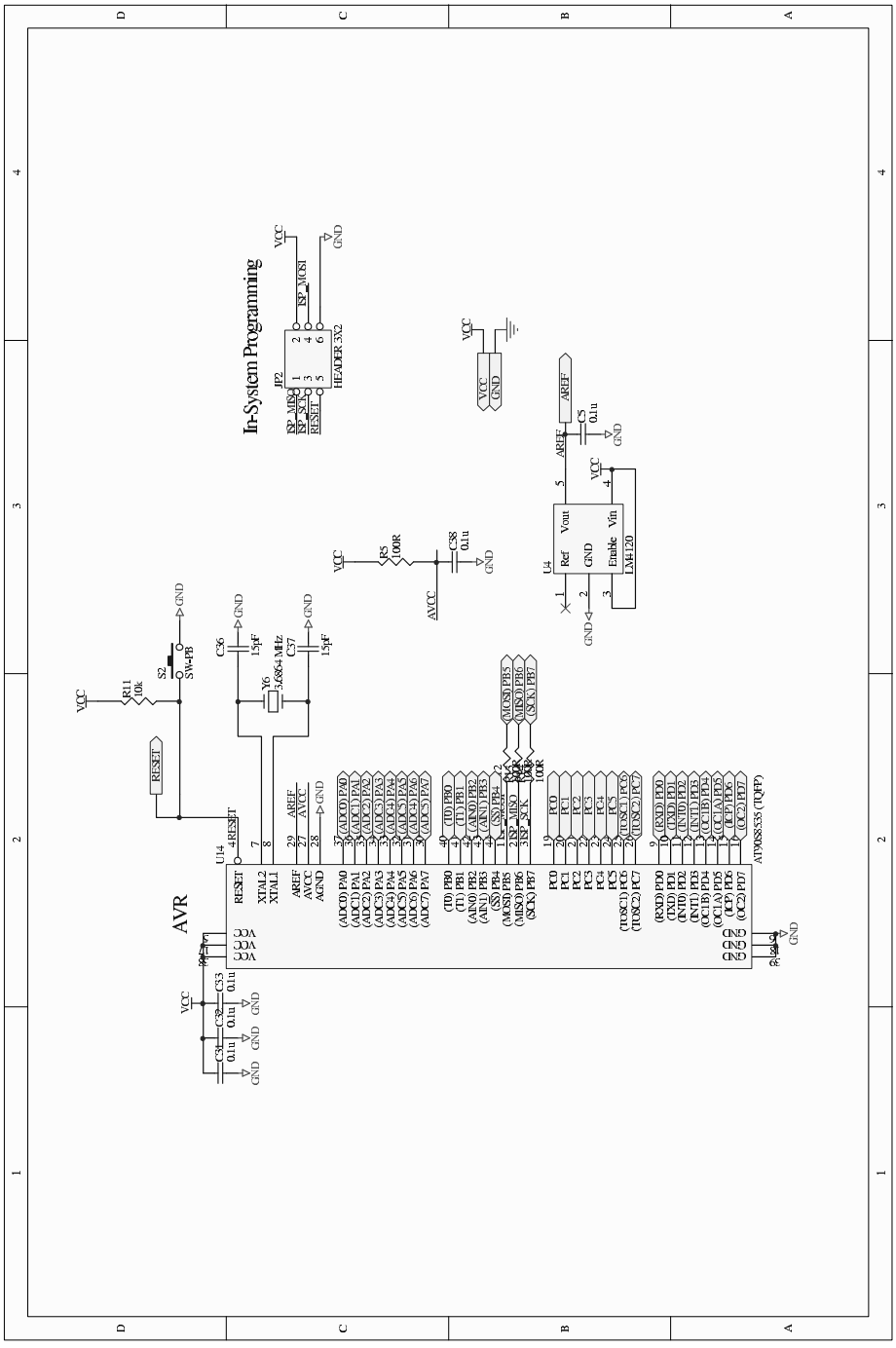


Figure A.3: Microcontroller module schematic

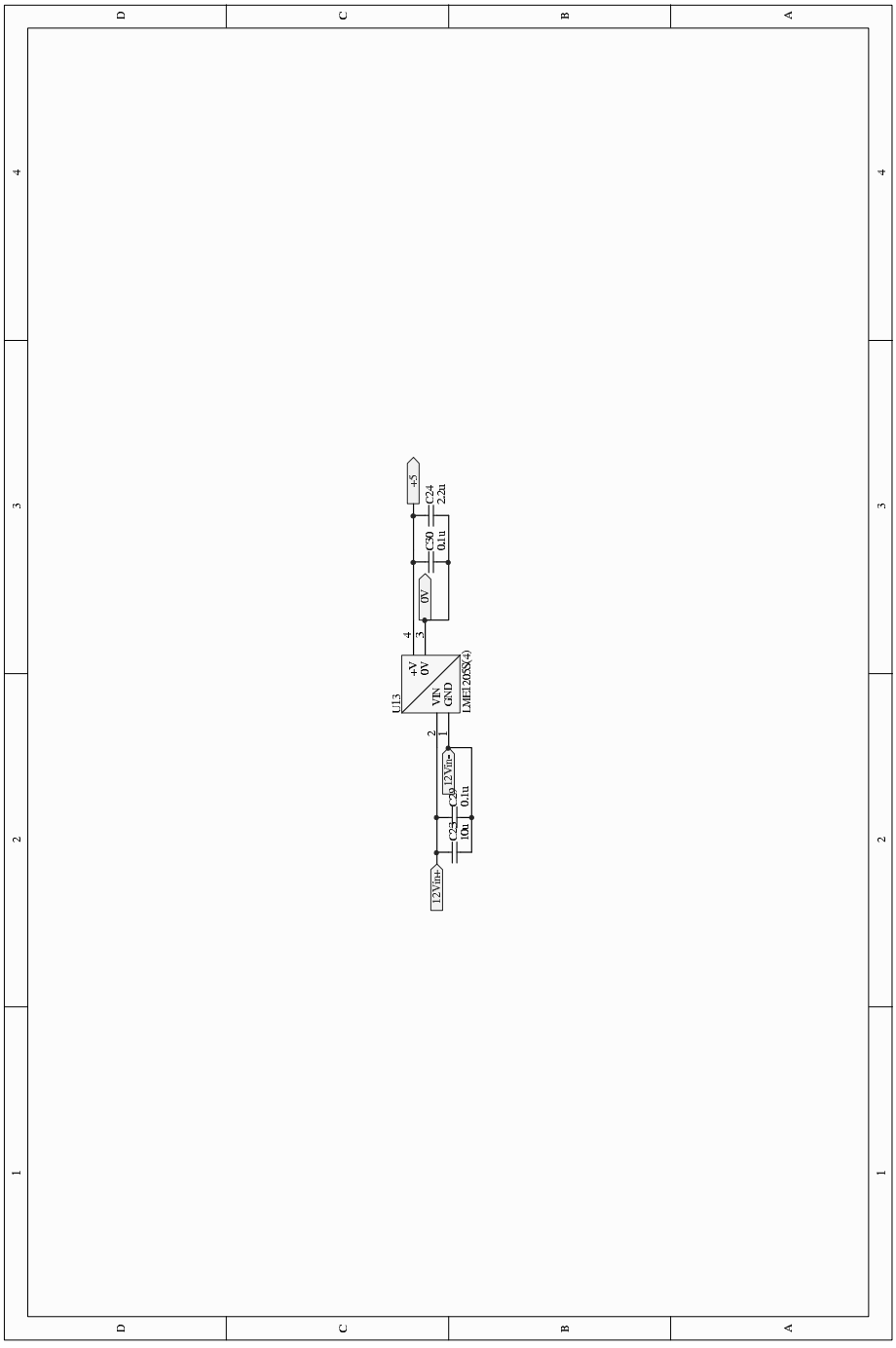


Figure A.4: DC-DC converter module schematic

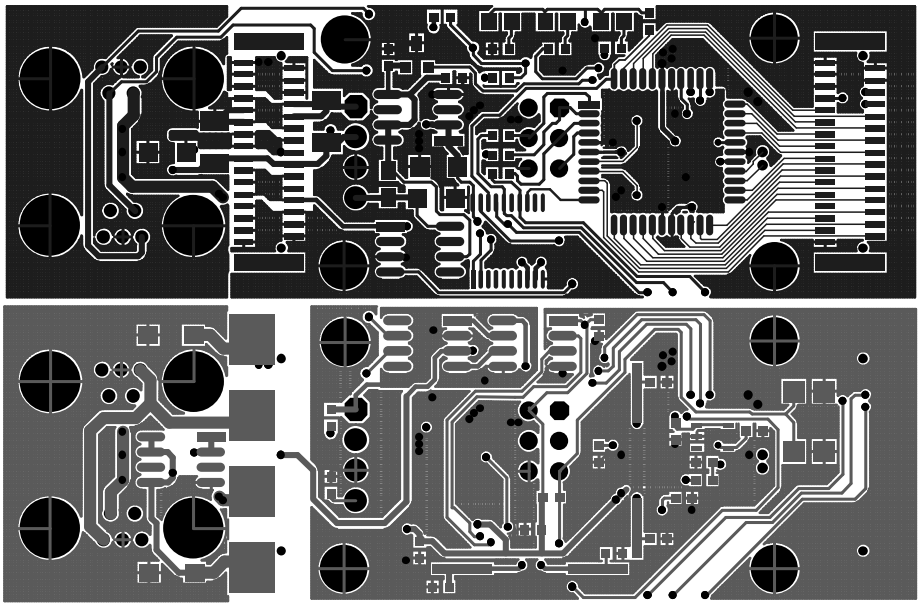


Figure A.5: CANRefNode artwork

Appendix B

Current Sensor Daughtercard Schematics

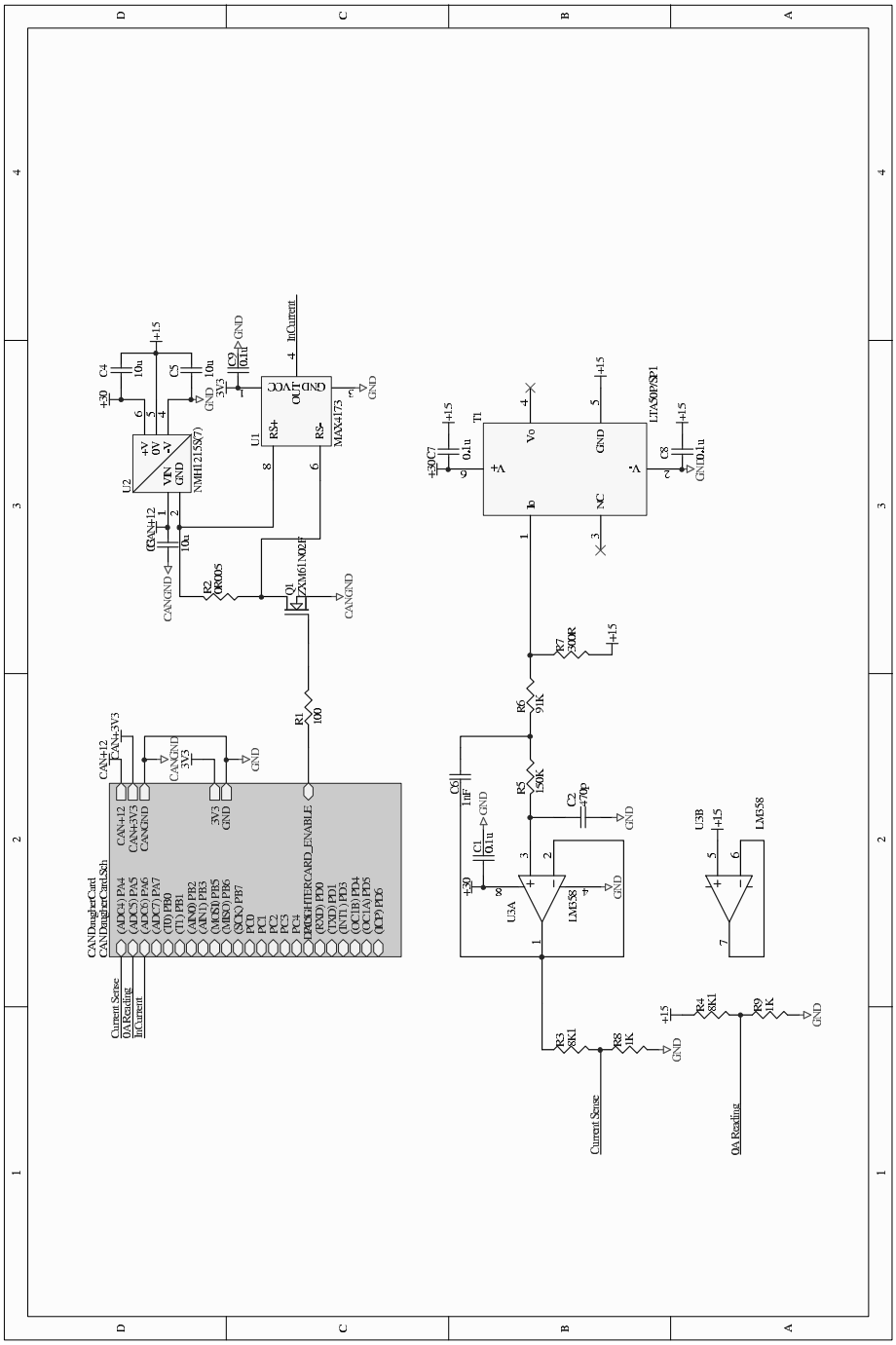


Figure B.1: Current sensor daughtercard schematic

Appendix C

Channels logged in Sunswift II

Sunswift II, 2001

Hydra
- BusVoltage
- BatteryCurrent
- MotorCurrent
- ArrayCurrent
- Speed
- ArrayTemp1
- ArrayTemp2
- BatteryTemp1
- BatteryTemp2
- BatteryTemp3
- BatteryTemp4

Sunswift II, 2002

MotorController
- PWM
- MotorCurrent
- ActualVelocity
- BusVoltage
- ControllerCurrent
- HeatsinkTemp
- MotorTemp
- ControllerTemp
- SMPTemp
- 15V
- MCHorn
- Brake
- MCRightIndicator
- MCLeftIndicator
- CurrentSetpoint
- VelocitySetpoint

DriverDisplay

BatteryCurrent
- Current
- Power
- AmbientTemp
- CurrentIntegration
- PowerIntegration

MotorCurrent
- Current
- Power
- AmbientTemp
- CurrentIntegration
- PowerIntegration

ArrayCurrent
- Current
- Power
- AmbientTemp
- CurrentIntegration
- PowerIntegration

DriverControls
- Horn
- LeftIndicator
- RightIndicators
- Camera

HornScreen
- Channel1Current
- Channel1Voltage
- Channel2Current
- Channel2Voltage
- AmbientTemp
- RefnodeCurrent
- Channel1Status
- Channel2Status

RBatFan
- Channel1Current
- Channel1Voltage
- Channel2Current
- Channel2Voltage
- AmbientTemp
- RefnodeCurrent
- Channel1Status
- Channel2Status

LBatFan
- Channel1Current
- Channel1Voltage
- Channel2Current
- Channel2Voltage

- AmbientTemp
- RefnodeCurrent
- Channel1Status
- Channel2Status

BLIndicator
- Channel1Current
- Channel1Voltage
- Channel2Current
- Channel2Voltage
- AmbientTemp
- RefnodeCurrent
- Channel1Status
- Channel2Status

BRIndicator
- Channel1Current
- Channel1Voltage
- Channel2Current
- Channel2Voltage
- AmbientTemp
- RefnodeCurrent
- Channel1Status
- Channel2Status

FrontIndicator
- Channel1Current
- Channel1Voltage
- Channel2Current
- Channel2Voltage
- AmbientTemp
- RefnodeCurrent
- Channel1Status
- Channel2Status

MPPT1
- PanelVoltage
- PanelCurrent
- OutputVoltage
- SenseVoltage
- HeatsinkTemp
- AmbientTemp
- TargetVoltage

MPPT2
- PanelVoltage
- PanelCurrent
- OutputVoltage
- SenseVoltage
- HeatsinkTemp
- AmbientTemp
- TargetVoltage

MPPT3
- PanelVoltage
- PanelCurrent
- OutputVoltage
- SenseVoltage
- HeatsinkTemp
- AmbientTemp
- TargetVoltage

MPPT4
- PanelVoltage
- PanelCurrent
- OutputVoltage
- SenseVoltage
- HeatsinkTemp
- AmbientTemp
- TargetVoltage

MPPT5
- PanelVoltage
- PanelCurrent
- OutputVoltage
- SenseVoltage
- HeatsinkTemp
- AmbientTemp
- TargetVoltage

MPPT6
- PanelVoltage
- PanelCurrent
- OutputVoltage
- SenseVoltage
- HeatsinkTemp
- AmbientTemp
- TargetVoltage

MPPT7
- PanelVoltage
- PanelCurrent
- OutputVoltage
- SenseVoltage
- HeatsinkTemp
- AmbientTemp
- TargetVoltage

DCDC
- BusVoltage
- HVCurrent
- UnproBusV
- ChassisTemp
- 12V
- 5V
- Unpro12V
- Unpro5V
- 12VCurrent
- 5VCurrent
- AmbientTemp

BatteryTemperatures
- Pack1Middle
- Pack1Edge
- Pack2Middle
- Pack2Edge
- Pack3Middle
- Pack3Edge
- Pack4Middle
- Pack4Edge

Appendix D

Graphs of Logged Data

The graphs presented in this appendix were generated from data logged on the 27th of October, 2002.

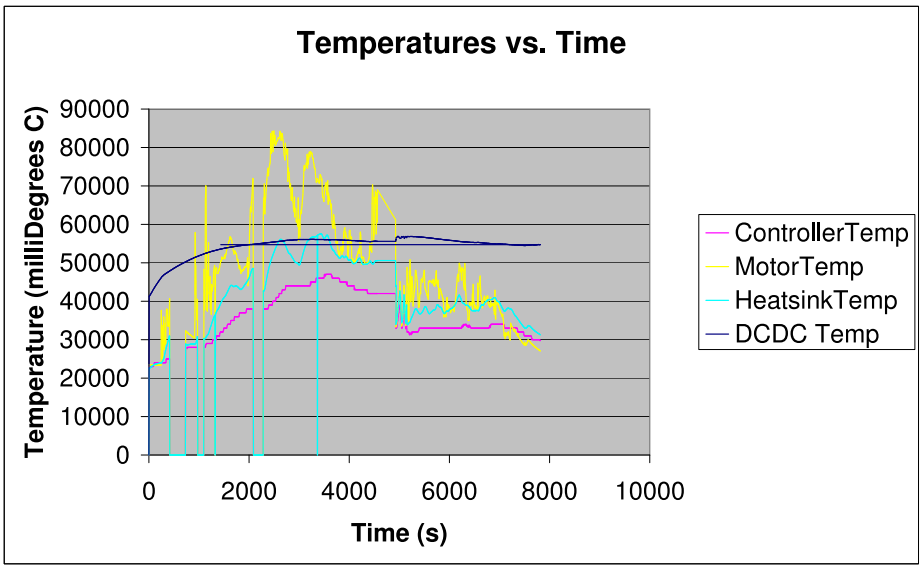


Figure D.1: Motor and DC-DC converter temperatures for testing day 6

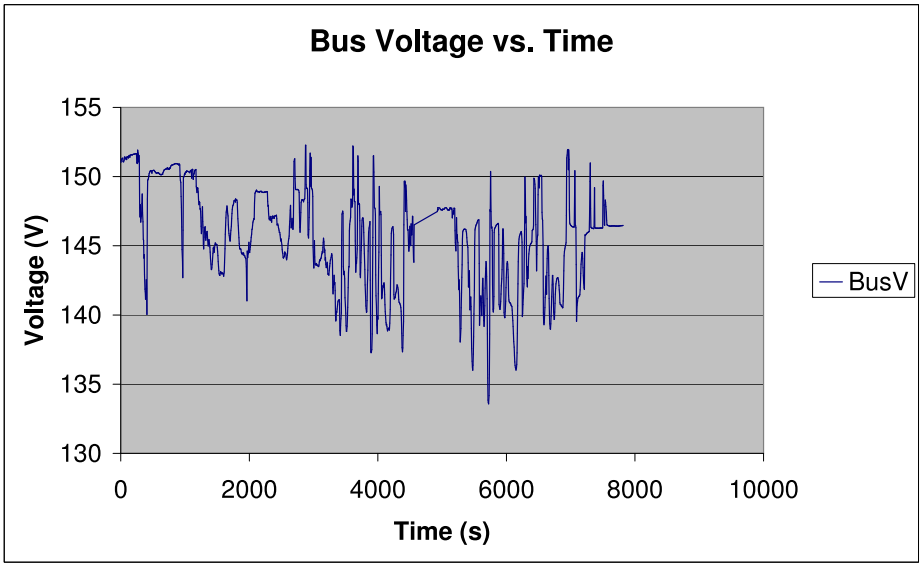


Figure D.2: Battery voltage vs. Time for testing day 6

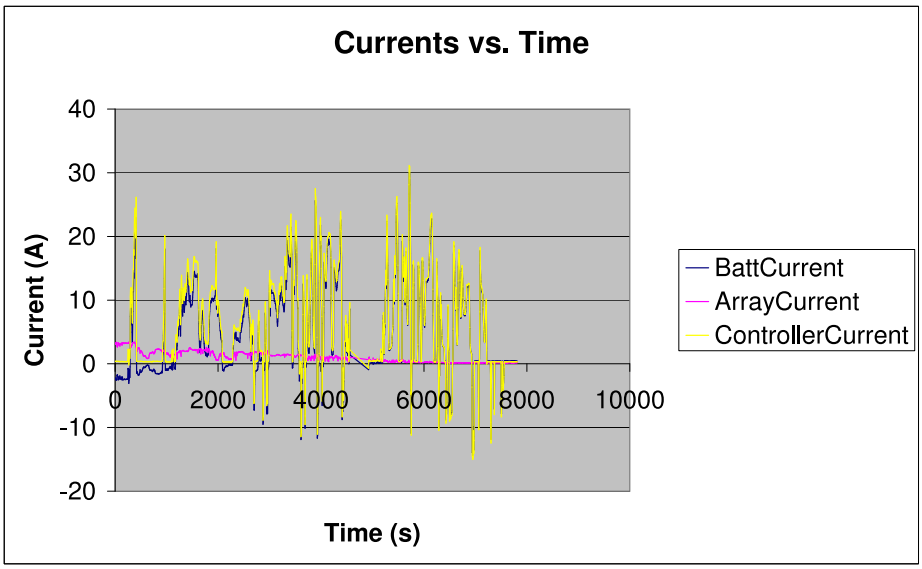


Figure D.3: Currents vs Time for testing day 6

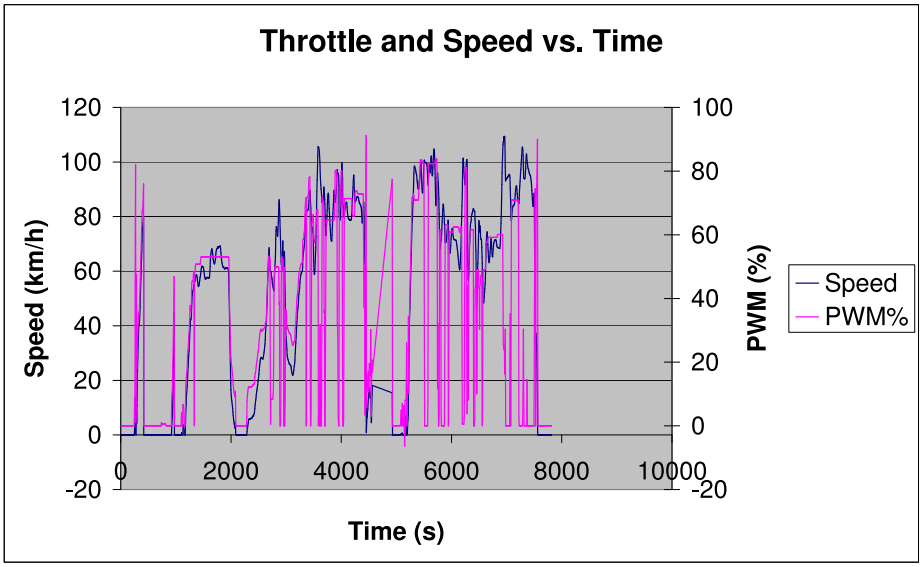


Figure D.4: Throttle and Speed vs. Time for testing day 6

Appendix E

Car Schematics

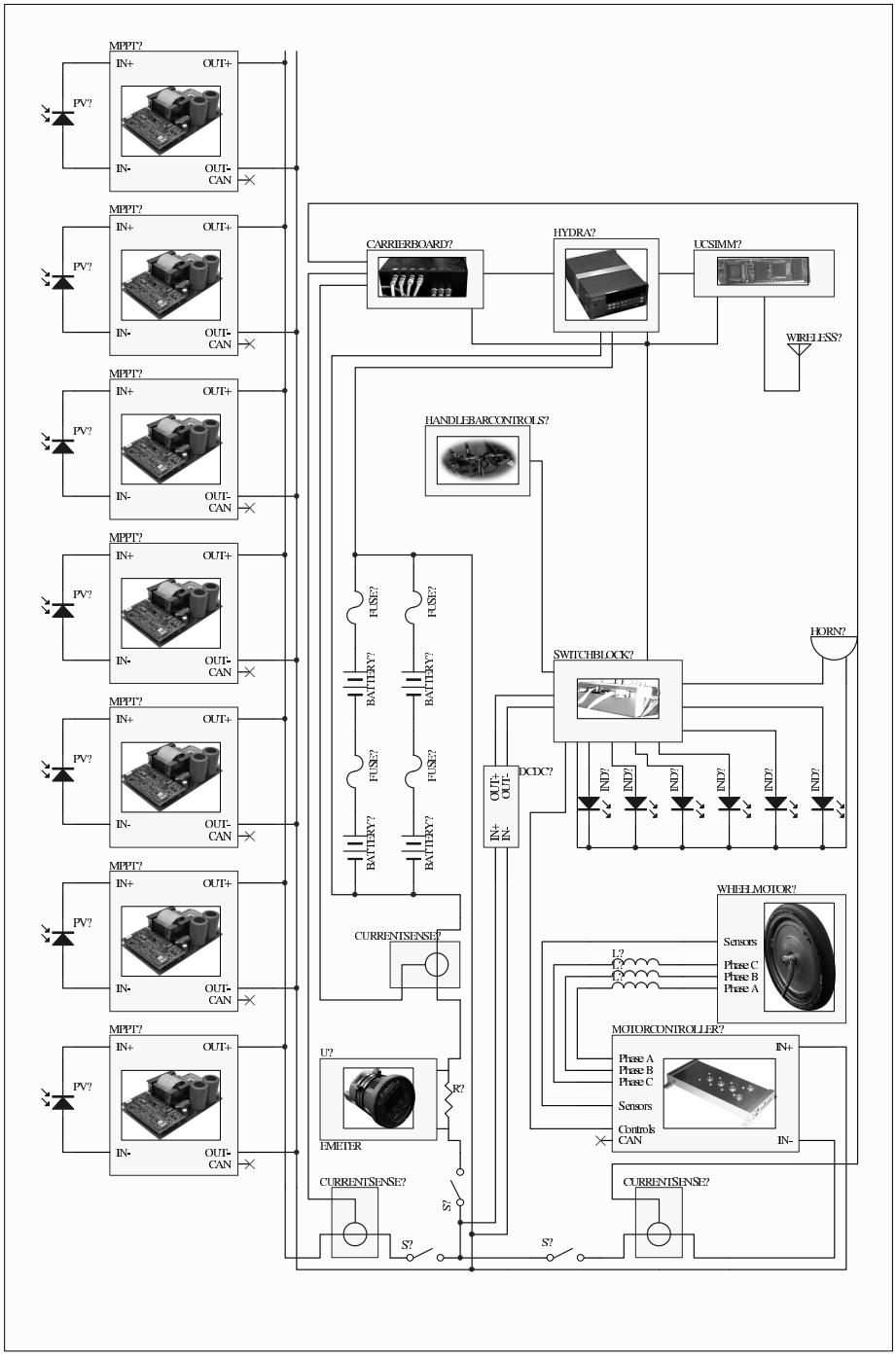


Figure E.1: Summary schematic of Sunswift II, 2001